SECURITY CLASSIFICATION OF THIS PAGE

DTIC FILE COPY (2)

REPORT DOCUMENTATION PAGE

AD-A196 100

1b. RESTRICTIVE MARKINGS

2b. DECLASSIFICATION / DOWNGRADING SCHEDULE

3. DISTRIBUTION / AVAILABILITY OF REPORT
unlimited

4. PERFORMING ORGANIZATION REPORT NUMBER(S)

5. MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION
The Regents of the University of California

6b. OFFICE SYMBOL *(If applicable)*

7a. NAME OF MONITORING ORGANIZATION
SPAWAR

6c. ADDRESS *(City, State, and ZIP Code)*
Berkeley, California 94720

7b. ADDRESS *(City, State, and ZIP Code)*
Space and Naval Warfare Systems Command
Washington, DC 20363-5100

8a. NAME OF FUNDING / SPONSORING ORGANIZATION
DARPA

8b. OFFICE SYMBOL *(If applicable)*

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

8c. ADDRESS *(City, State, and ZIP Code)*
1400 Wilson Blvd.
Arlington, VA 22209

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.

11. TITLE *(Include Security Classification)*
* THE ANALYSIS OF DISKLESS WORKSTATION TRAFFIC ON AN ETHERNET

12. PERSONAL AUTHOR(S)
* Riccardo Gusella

13a. TYPE OF REPORT
technical

13b. TIME COVERED
FROM ________ TO ________

14. DATE OF REPORT *(Year, Month, Day)*
* November, 1987

15. PAGE COUNT
* 49

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

Enclosed in paper.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION
unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL

22b. TELEPHONE *(Include Area Code)*

22c. OFFICE SYMBOL

83 APR edition may be used until exhausted.
All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

Productivity Engineering in the UNIX† Environment

The Analysis of Diskless Workstation Traffic on an Ethernet

Technical Report

S. L. Graham
Principal Investigator

(415) 642-2059

"The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government."

Contract No. N00039-84-C-0089

August 7, 1984 - August 6, 1987

Arpa Order No. 4871

†UNIX is a trademark of AT&T Bell Laboratories

The Analysis of Diskless Workstation Traffic on an Ethernet

Riccardo Gusella

Computer Systems Research Group
Computer Science Division
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
Berkeley, CA 94720

ABSTRACT

This study analyzes the communication traffic on a medium-size Ethernet local area network that connects file servers to diskless workstations. Our measurements differ from those of other studies in two important aspects. First, the Ethernet traffic is much higher than the traffic measured in earlier studies. Second, the measurements are more accurate than any previous ones. For each packet, all the protocol information was extracted and recorded on tapes along with the packet arrival time. Less than one percent of the total number of packets was lost. The clock used to timestamp the packet records had a one-microsecond resolution.

In this paper we describe the measurement methodology, present the traffic statistics, and compare our traffic characteristics with those reported in a 1979 study by Shoch and Hupp. We detail the behavior of each of the protocols responsible for a significant proportion of the total traffic. Our study suggests that protocol behavior is sub-optimal. Traffic patterns are highly skewed: a single client workstation, under normal usage, can generate for and request from a file and paging server data amounting to more than 20 percent of the total raw Ethernet bandwidth. In addition, protocol timers cause numerous, unnecessary retransmissions.

The accurate tracing time has allowed us to study extensively the packet interarrival time and has provided strong grounds for the hypothesis that the arrival process is non-random. Based on projections from our analyses, we conjecture that network bandwidth may be a scarce resource in applications that involve high-performance virtual-memory diskless machines. We believe that a key prerequisite for the success of future diskless workstations will be the design of fast interactive communication protocols that function effectively under high network load.

1. Introduction

With the development and increasing use of distributed systems, computer communication traffic over local area networks has changed. Conventional remote terminal access and file transfer are no longer the dominant applications. More and more computing systems are built

The measurements described and analyzed in this paper were taken at the Computer Science Division of the University of California, Berkeley. The trace analysis was performed while the author was visiting the Computing Science Research Center of AT&T Bell Laboratories.

This work was sponsored in part by AT&T Bell Laboratories, in part by CSELT S.p.A. of Italy, and in part by the Defense Advanced Research Projects Agency (DoD), Arpa Order No. 4871 monitored by the Space and Naval Warfare Systems Command under contract No. N00039-84-C-0089.

around diskless workstations and file servers. These workstations are economical, expandable, easy to manage, and more suitable for offices, where a large disk subsystem would produce noise and heat. Often they provide a virtual memory environment and need to page to a remote paging device. Whereas older protocols only had to respond to the slow, interactive terminal traffic and provide throughput for the bulk transfer of data, newer ones have to provide much faster response time for file access and quick network access to the backing store.

As a result of modifications aimed at increasing the performance of computer systems, now a single packet often only contains a portion of the object being transported. For instance, at the time the Ethernet was designed, file pages were 512 bytes or 1K bytes in size;[54, 55] now, 4K bytes appears to be the standard. A SUN3's memory page is 8K bytes, which is transferred to the paging device in a single transaction during paging or swapping. Additionally, new programming interfaces have increased users' need for bandwidth. Dumping the image of a graphic display may require the transfer of millions of bits. In the case of diskless workstations, these transactions are performed through network protocols, which, at the physical level, involve the transmission of a number of *frames* in quick sequence.

The Ethernet, designed at the Xerox Palo Alto Research Center (PARC) at the beginning of the seventies[17] and subsequently modified for higher speed,[15] is one of the most successful local area networks. It combines simplicity and low cost with reasonably high bandwidth. The characteristics of its traffic have been repeatedly studied starting with basic measurements performed about eight years ago at Xerox PARC by J.F. Shoch and J.A. Hupp.[51] Shoch and Hupp suggested, and it is now commonly accepted,[27] that the Ethernet under "normal" workloads is lightly loaded and that hosts and host interfaces, rather than the network, are the protocols' performance bottlenecks. As we shall see, the developments in distributed systems and user interfaces have substantially modified the characteristics of the Ethernet traffic.

These developments require a re-evaluation of network and protocol performance. This paper describes and analyzes traffic measurements, taken at the University of California at Berkeley (UCB), aimed at determining the performance of an Ethernet local area network in the presence of file system access performed by diskless workstations. An essential part of the analysis is the interpretation of network communication protocol behavior.

The next section describes the Berkeley user community and the specific network studied. Section 3 describes the measurement methodology; an important goal of this study was to design a measurement system that would generate the most accurate picture of the traffic possible. Section 4 presents the general statistics of the measured traffic and compares our traffic with the traffic that was described eight years ago by Shoch and Hupp. Since we contend that it is not possible to describe and understand the network behavior without careful analysis of its protocols, Sections 5, 6, and 7 examine in detail the three protocols that transported the largest amount of data: the Transmission Control Protocol as part of the Defense Advanced Research Projects Agency (DARPA) family of protocols, the SUN Network Disk protocol, and the SUN Network File System protocols. In this, we depart from the analysis of Shoch and Hupp, who were solely concerned with general Ethernet properties and statistics. (We describe the protocols' features in the context of data analysis; our descriptions are, for lack of space, necessarily incomplete. When necessary, the references will provide full details on the subject.) Each section concludes with a summary of the main data and our interpretations. Finally, in the last two sections we compare our results with those of projects more recent than the Xerox PARC measurements and we present the conclusions of our study.

2. The Berkeley User Community and the XCS Network

By and large the UCB computer science user community is similar to those of other universities and large research organizations.† User activities include program development, text editing, experimentation with computation-intensive programs, and operating system research, which

† The description refers to the Computer Science user community at the University of California at Berkeley and its facilities at the time of the measurements in Spring 1986.

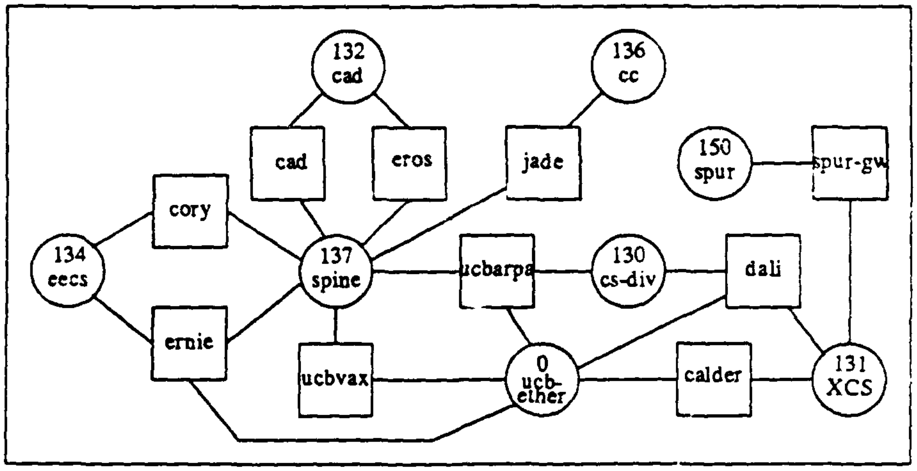
frequently requires rebooting machines.

The UNIX† operating system is by far the most common environment in use. It includes UNIX 4.3BSD and UNIX 4.2BSD developed by the Computer System Research Group, SUN UNIX, which is based on 4.2BSD, and DEC ULTRIX, also derived from 4.2BSD. Berkeley UNIX 4.2BSD includes an implementation of the DARPA family of protocols based on the Transmission Control Protocol (TCP)[45] and the Internet Protocol (IP).[46] Berkeley UNIX 4.3BSD, a subsequent release, adds the Xerox Network Systems (NS) family of protocols.[16] SUN UNIX includes an implementation of the Remote Procedure Call protocol (RPC)[36] as well as two protocols to support SUN's file system implementation: the Network Disk protocol (ND)[37] and the Network File System protocol (NFS).[38]

Although most professors and some students own workstations (mainly SUN, MicroVAX, and Xerox Star workstations),†† many students still use standard terminals. Most students share offices and their equipment. Therefore, workstations in student offices, unlike those owned by professors, tend to be in use during most of the day. While most diskless workstations are located in professors and students' offices, those in the xcssun cluster are in a public room accessible by those students who do not have any workstation in their offices.

Most users read their electronic mail on three VAX's: ucbarpa, ernie, and kim. Often workstation users use a window to connect to one of these machines and therefore generate character traffic from the workstation to the VAX's.

FIGURE 2-1. UCB NETWORKS AND GATEWAYS



There are a large number of local area networks, mostly Ethernets. A portion of the traffic on each independent network is directed externally, towards other local area networks or wide area ones, most notably to the Berkeley Arpanet Interface Message Processor (IMP). Figure 2-1 displays a subset of the networks in the UCB campus: those that are most often used by Computer Science Division users. In this map, circles represent local area networks, and boxes represent gateways. (Not all the gateways are represented in the figure.) The networks are identified by both their subnetwork number‡ and by their symbolic name. All of them are 10 Mb/s Ethernets except for *ucb-ether*, which is a 2.94 Mb/s Ethernet. All gateways, except for spur-gw, which is a SUN workstation, are VAX's. Networks 0, 130, 131, and 150 belong to the Computer Science

† UNIX is a trademark of AT&T Bell Laboratories.

†† Both MicroVAX's and Xerox Star machines have local disks.

‡ The host part of class B IP addresses, which are divided into network and host parts (two 16-bit quantities), is, in 4.3BSD, further divided into two 8-bit fields: a local area subnetwork number and a host number.[28]

Division. Networks 132 and 136 belong respectively to the EECS Department and the Computer Center of the university. Network 134 is used by the administration of the Electrical Engineering and Computer Sciences Department; and network 137 functions as a bridge among many otherwise separate local area networks.

The Ethernet we chose to study, a single-cable Ethernet which UCB system administrators have denominated *XCS* network, covers three floors of Evans Hall, from the fourth to the sixth, where professors and students have their offices. The network, which is also numbered as subnetwork 131 in Figure 2-1, connects about a hundred machines: 41 diskless SUN workstations arranged in clusters around six file servers; 23 XEROX Star workstations; three VAX's, two of which, *calder* and *dali*, act as *gateways*, i.e. hosts that interconnect networks; several MicroVAX's; and a few Symbolics and TI Explorer LISP machines. There is a third gateway: a SUN workstation, *spur-gw*, between the spur network and XCS. Table 2-1 shows the six *clusters* of SUN workstations. (The machine names appear as they are spelled [or misspelled!] at Berkeley.)

There are three machine rooms, one on each floor The machine we used for the measurements is on the fourth floor; the SUN file servers *dim*, *snow* and *xcssun* are on the fifth floor; while the other three file servers *sequoia*, *lassen*, and *teak* are on the sixth.

3. The Measurement Methodology

We constructed a measurement system that did not affect the network under analysis. For the data collection we used a large machine, a VAX 8600 with 32M bytes of physical memory, which was only lightly loaded. This VAX, *vangogh*, had two network interfaces: a DEC DEUNA connected to the XCS network, and an Interlan interface connected to subnetwork 130 (the *cs-div* network). We configured the kernel to route output messages to the Interlan interface and to ignore the other interface. Therefore, the traffic generated by the few occasional users and by the various Berkeley UNIX daemons did not affect either the XCS network or the DEUNA network interface, which was programmed to act only as a receiver of all packets. No other machine on the Berkeley campus was informed of the existence of the network address corresponding to the DEUNA interface. Thus, no traffic on the XCS network was directly addressed to vangogh. (Vangogh was, however, reachable on subnetwork 130.)

The DEUNA interface uses internal buffers to hold received packets until they are transferred using DMA to the host buffer memory. Vangogh, informed by an interrupt of the successful transfer, extracted the Ethernet packet header and the portion of the data that contained protocol information, and time stamped the record with a microsecond accurate clock.

The DEUNA driver employed a double buffering technique: it kept two large buffers in kernel memory where it stored the packet information that it was collecting. When the first buffer was full, it switched to the second one while a sleeping kernel process was awakened to asynchronously transfer the data in the first buffer to a trace file on disk. This process alternated between the two buffers.

When a trace file reached the size of 130M bytes– this size was chosen because it fit in one standard tape at 6250 bpi, and three files fit on one disk drive dedicated to the measurements– the kernel driver automatically switched to another one. Dumping the data onto tape presented no problem since it took only four minutes. In contrast, it took about eight hours to fill up one file during the night and about a hundred minutes at the daily peak of network activity during the late afternoon.

An important goal of our measurements was to reduce the packet loss to a minimum. We were able to know how many packets were lost by reading counters that the DEUNA maintains in its internal memory. These include a counter for the number of packets received, a second for the number of packets lost due to internal buffer overflow, and a third for the number of packets lost due to host buffer overflow. The internal buffer can overflow when the data arrival rate over the Ethernet exceeds the DMA transfer rate over the UNIBUS. (Notice that this is possible since the Ethernet bit rate of 10 Mb/s corresponds– taking into account header overhead, preamble generation time, and interpacket spacing– to a maximum rate of 1,220K bytes per second when all

TABLE 2-1. CLUSTERS OF SUN WORKSTATIONS †

FILE SERVER	CLIENT WORKSTATIONS	CPU TYPE	MEMORY SIZE	ETHERNET INTERFACE
dim		SUN2	3M bytes	3Com
	chip	SUN2	2M bytes	3Com
	mike	SUN2	4M bytes	Sun Intel
	rob	SUN2	2M bytes	3Com
lassen		SUN2	4M bytes	Sun Intel
	baobab	SUN2	2M bytes	3Com
	pawlonia	SUN2	4M bytes	Sun Intel
	pine	SUN2	2M bytes	Sun Intel
	maple	SUN2	4M bytes	Sun Intel
	rip	SUN2	4M bytes	Sun Intel
	sleepy	SUN2	2M bytes	3Com
sequoia		SUN3	4M bytes	Sun Intel
	ash	SUN3	4M bytes	AMD Am7990 LANCE
	elm	SUN2	4M bytes	Sun Intel
	fir	SUN2	4M bytes	3Com
	liveoak	SUN2	4M bytes	Sun Intel
	madrone	SUN3	4M bytes	AMD Am7990 LANCE
	mahogany	SUN3	4M bytes	AMD Am7990 LANCE
	oak	SUN2	4M bytes	Sun Intel
	redwood	SUN2	4M bytes	Sun Intel
	shangri-la	SUN3	4M bytes	AMD Am7990 LANCE
	walnut	SUN2	4M bytes	Sun Intel
snow		SUN2	2M bytes	3Com
	bashful	SUN2	2M bytes	3Com
	doc	SUN2	2M bytes	3Com
	dopey	SUN2	2M bytes	3Com
	grumpy	SUN2	2M bytes	3Com
	happy	SUN2	2M bytes	3Com
	pinocchio	SUN2	2M bytes	3Com
	sneezy	SUN2	2M bytes	3Com
teak		SUN2	4M bytes	Sun Intel
	bansai	SUN3	4M bytes	AMD Am7990 LANCE
	eucalyptus	SUN3	4M bytes	AMD Am7990 LANCE
	geppetto	SUN3	4M bytes	Sun Intel
	ginko	SUN3	4M bytes	AMD Am7990 LANCE
	larch	SUN2	4M bytes	Sun Intel
	palm	SUN3	4M bytes	AMD Am7990 LANCE
	yew	SUN3	4M bytes	AMD Am7990 LANCE
xcssun		SUN2	2M bytes	3Com
	jupiter	SUN2	2M bytes	3Com
	mars	SUN2	2M bytes	3Com
	mercury	SUN2	2M bytes	3Com
	neptune	SUN2	2M bytes	3Com
	pluto	SUN2	2M bytes	3Com
	saturn	SUN2	2M bytes	3Com
	uranus	SUN2	2M bytes	3Com
	venus	SUN2	2M bytes	3Com

packets transmitted are of maximum length, while the maximum UNIBUS data transfer rate in a VAX equals to about 800K bytes per second.) In addition, packets can also be lost because the host may be slow in emptying the host buffer.

During the development and testing of our driver, we noticed that the number of lost packets decreased as we allocated more memory to the host buffer pool. By allocating space sufficient for storing 55 packets, we were able to reduce the percentage of packets lost because of internal buffer

† Those who wonder what happened to the missing planet in the xcssun cluster will be relieved to know that earth was on loan for Smalltalk development and was "orbiting" on the spur network. We shall not comment on the fact that pinocchio is among snow's dwarfs.

overflow to less than one percent of the number of packets received, and the fraction of packets lost because of host buffer overflow to less than one in 30,000.

The DEUNA interface, like most other Ethernet interfaces, only reports the number of collisions that occur as it tries to transmit packets.† Our DEUNA was programmed to receive all packets and vangogh never used it to transmit. Consequently, it was not possible to record on the traces the packet collisions as they occurred.

We started the measurement phase of the project by first performing a feasibility study on a SUN workstation set up in stand-alone mode to continuously monitor the traffic and display traffic statistics. This phase took about two weeks; two more weeks were needed to modify the DEUNA device driver and test the changes. The measurements themselves spanned a period of three weeks during April and May 1986. We recorded a total of about 6,500M bytes of data.

In this paper we have chosen to analyze a typical weekday for the XCS traffic: 24 hours, starting at 1 am, Tuesday May 6, 1986. Since for speed considerations the packet filter in the DEUNA device driver was extremely simple, we have processed the original traces to arrange the various protocol information in a uniform way before performing the data analysis. At the same time, by subtracting from the packet arrival time the DMA transfer time and the Ethernet transmission time, we have reconstructed the times when each packet transmission started, i.e. the times when the status of the Ethernet channel changed from idle to busy due to a packet transmission.

Although vangogh's clock was very accurate, the reconstructed times are subject to the variability of the DMA, which depends on the UNIBUS utilization; to the queuing time in the DEUNA, which depends on the packet arrival rate; and to the interrupt response time and the queuing time on the host. During the development of the measurement system, in order to assess the accuracy of our timing information, we programmed a workstation to transmit packets, which vangogh recorded, at regular intervals. (For greater accuracy, the workstation kernel rather than a user process sent these messages.) We reconstructed the sending times from the traces and computed the standard deviation of the estimated inter-packet times. As expected, the standard deviation was minimal during low network utilization, but as high as two milliseconds at high network load. However, this relatively large variance was due to a small number of arrivals that were delayed because of two factors: 1) the transmission of some of the packets was deferred because the channel was busy; and 2) the transmission was rescheduled because a collision occurred. Therefore, if we plotted the interarrival times of the packets in the experiment, a graph with a narrow and sharp peak would result.

4. General Statistics

The 24-hour trace under study consists of 11,837,073 records– one for each packet. The total number of bytes transmitted is 7,060,163,648 (this figure includes header, checksum, and data bytes). Although the mean network utilization– the proportion of time in which one transmitter was active– is a mere 6.5 percent, during the period of high traffic in the late afternoon, it is very common to have relatively long intervals of mean utilization ranging around and above 30 percent. By contrast, the 2.94-Mb/s Ethernet measured by Shoch and Hupp[51] – a network with about 120 machines– carried less than four percent of the total number of bytes we observed. Therefore, the first, most significant difference between our data and previous measurements is the higher utilization of the communication channel.

As observed before, the packets lost by our network interface amount to less than one percent. Most previous measurements projects[5, 14] suffered from much higher packet loss, in the range of 10 to 15 percent and more. Furthermore, the timing information in our traces is very accurate, which enables us to study the interarrival times extensively. Therefore, the second critical difference is the accuracy of the data gathered.

† We assume that our readers are familiar with the principles of Carrier Sense Multiple Access with Collision Detection (CSMA/CD) of Ethernet.

4.1. Utilization

Figure 4-1 displays the network utilization in the 24-hour period under study. Each vertical line represents the fraction of time during a one-minute interval in which a transmitter was active. Thus, the packet headers, the data bytes, and the packets' checksums, as well as the 8-byte preamble, which is transmitted prior to each packet to synchronize the receiver and the transmitter, are all counted in calculating the utilization. Notice, however, that we do not take into account the bandwidth wasted on collisions. Although not supported by actual measurements [as we observed in Section 3, our network interface did not report the number of collisions], we believe that there are two factors that keep the number of collisions low. First, as we shall see in the following sections, each station generates packets in bursts that are followed by relatively long silent periods. Second, the interval between packets in a burst is much larger than the Ethernet minimum interframe spacing, which decreases the chances that packets from another burst will collide with ones from the first.

FIGURE 4-1. ETHERNET UTILIZATION (ALL PACKETS)

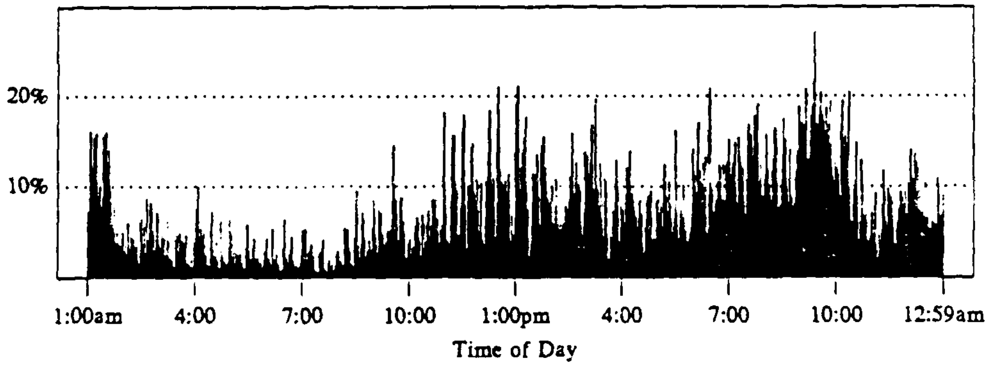
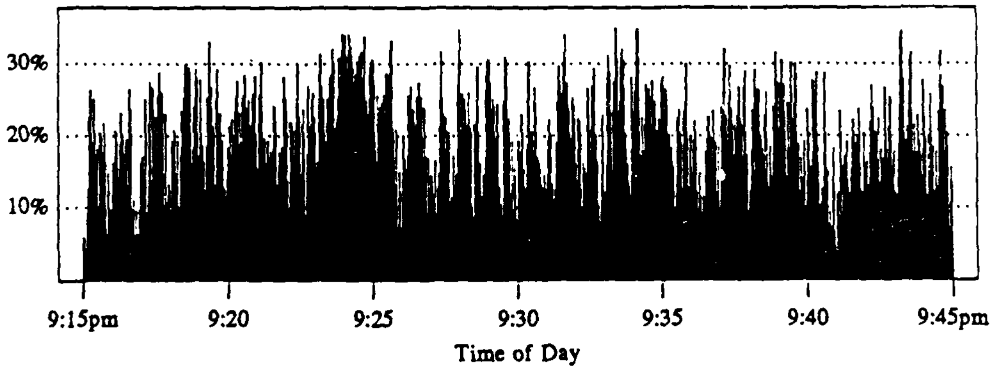


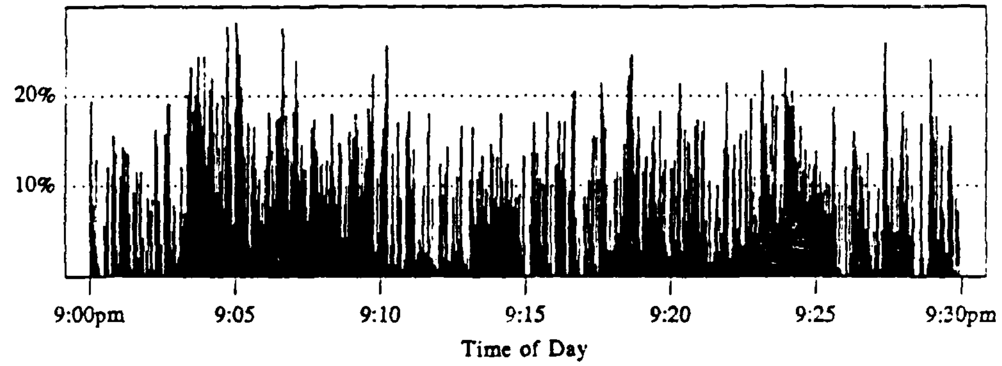
Figure 4-1 shows that the utilization, measured over one-minute intervals, rarely exceeds 20 percent. If, however, we examine the utilization more closely over smaller intervals, we see that in fact the Ethernet behavior is quite bursty and frequently reaches levels over 30 percent. This is shown in Figure 4-2, which displays the network utilization from 9:15 pm to 9:45 pm over one-second intervals.

FIGURE 4-2. ETHERNET UTILIZATION (ALL PACKETS)



This traffic would overload Shoch and Hupp's 2.94-Mb/s Ethernet, and would also saturate the IBM 4-Mb/s ring network. Furthermore, we found that hardware changes– faster computers and more intelligent network interfaces– raised the utilization rate. For instance, the communication between just two computers, the SUN file server *sequoia* (a SUN3/180) and the client workstation *madrone* (a SUN3/50), can load the Ethernet more than 20 percent and displays peak transfer rates as high as 275K bytes per second, as shown in Figure 4-3.†

FIGURE 4-3. ETHERNET UTILIZATION (TRAFFIC BETWEEN SEQUOIA AND MADRONE)



This behavior has been observed in experimental environments where two machines sent artificially generated data to each other.[14,29] However, its presence in a real-world environment compels us to re-evaluate the notion that under normal traffic an Ethernet is lightly loaded.[51] Unless protocol and system designers explicitly take into account message latencies generated by higher loads, they may develop facilities that behave sub-optimally. For example, if protocol timers are set to a value that is too small in the presence of network contention, they could expire prematurely, before an acknowledgement or a response packet is received, leading to unnecessary message retransmission.

4.2. Packet Length

Figure 4-4 and Figure 4-5 display respectively the number of packets and number of bytes transmitted as a function of the packet length.††

In Shoch and Hupp's Ethernet the packet size distribution was bimodal and most of the traffic volume was carried by a small number of large packets; the mean and median packet lengths were 122 and 32 bytes respectively. In our network, based on a modern environment of diskless workstations with virtual memory operating systems, the mean packet size is 578 bytes and the median is 919 bytes. However, since the maximum packet sizes of the two Ethernets are different– 554 for the 2.94-Mb/s and 1500 for the 10-Mb/s Ethernet– a more revealing comparison of the these values can be obtained by dividing each mean and median by the corresponding maximum packet length. In the case of the Xerox PARC Ethernet the mean is 22.0 percent, and the median 5.8 percent, of the maximum packet size; in our Ethernet, the mean is 38.5 percent and the median 61.3 percent.

In Figure 4-4 and 4-5 we can identify three areas: the region near the origin, mainly determined by the small packets generated by the interactive terminal traffic and the protocol's acknowledgements of TCP; the region in the middle, characterized by the parameter-passing packets of the RPC-based NFS protocol; and the region on the right, which accounts for most of the packets and most of the bytes, and whose components are, in decreasing order, ND, NFS, and TCP.

The smallest packet sizes,‡ those less than 50 bytes, together transport 30.4 percent of the

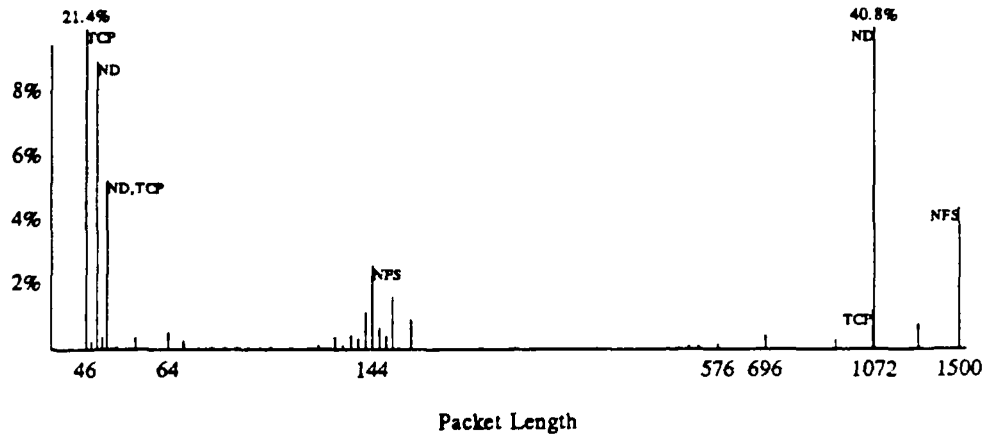
† Madrone's users were developing a language-based editor in a LISP environment. The editor is written in Franz Lisp for the most part, although it includes a large component of the SunView windowing mechanism. The size of the object file is around 3M bytes. Therefore, madrone compiled both LISP and C programs, and ran big LISP processes within the SUN window manager. Often large LISP images were dumped.[4]

†† In the remainder of this section, when we mention packet lengths, we refer to the length of the data portion of the Ethernet packet, i.e. the full packet length minus 14 header bytes and 4 checksum bytes.

‡ The minimum size packet transported by the 10-Mb/s Ethernet is 46 data bytes.

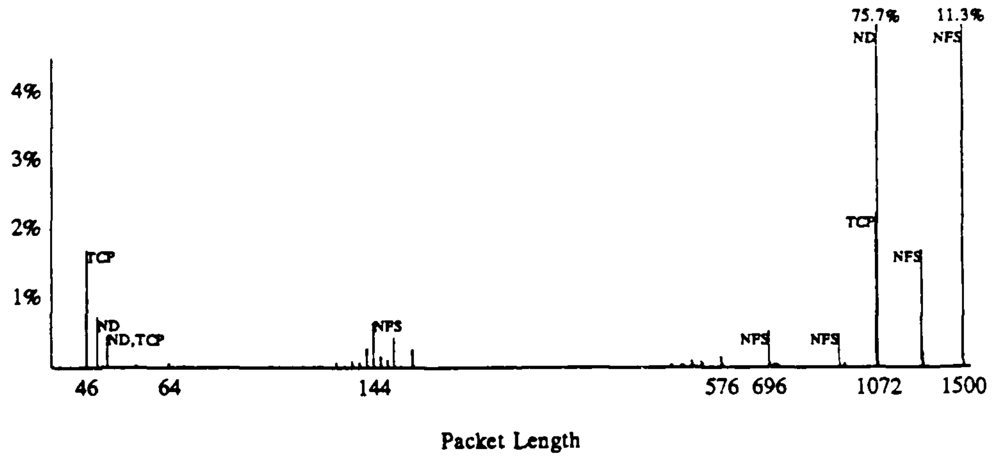
total packets but only 2.9 percent of the total bytes. A notable portion, 30.8 percent, of these packets is used to carry requests for ND protocol transfers. Yet 66.8 percent of them are used by the character and acknowledgement traffic of TCP.

FIGURE 4-4. PERCENTAGE OF PACKETS VS. PACKET LENGTH



The packets whose lengths are between 124 and 168 bytes are used almost exclusively by the NFS protocol, which is based on *request-response* remote procedure calls. They account for 8.3 percent of the total packet count, but for only 2.2 percent of the total byte count. We believe that this segment of packet lengths will acquire more and more weight in the future as applications based on request-response protocols develop.[14, 48]

FIGURE 4-5. PERCENTAGE OF BYTES VS. PACKET LENGTH



Finally, the packets whose lengths are larger than 576 bytes comprise 50.2 percent of the total packet count and 93.1 percent of the byte traffic. TCP transports 1.4 percent of the packets and 2.6 percent of the bytes. ND and NFS transport 40.8 and 6.2 percent of the packets and 75.7 and 14.6 percent of the bytes respectively.

4.3. Network Protocols

Table 4-1 lists the number of packets and bytes transferred and the mean packet size for each of the major protocols in use on XCS. The entries in the *DATA BYTES* column of Table 4-1 are computed by summing up the lengths of the data field in the Ethernet packets. The mean packet sizes in the last column are instead computed from the full packet lengths. We observe again that the remote file system protocols, ND and NFS, have relatively large mean packet sizes and comprise most of the bytes transferred, whereas the conventional character traffic of TCP has a relatively small mean packet size.

TABLE 4-1. PROTOCOL STATISTICS

PROTOCOL	PACKETS		DATA BYTES		BROADCAST	MEAN
	NUMBER	PERCENTAGE	NUMBER	PERCENTAGE	PACKETS	PACKET SIZE
nd (ip)	6,158,602	52.03	5,245,577,140	76.61	323	869.75
nfs (ip)	1,880,402	15.89	1,179,797,526	17.23	0	645.42
tcp (ip)	3,379,564	28.55	356,640,214	5.21	0	123.52
udp (ip)	111,553	0.94	36,474,286	0.53	17,182	344.97
ns	133,491	1.13	18,791,982	0.27	20,681	158.77
arp	110,705	0.94	5,098,296	0.07	108,830	64.05
icmp (ip)	41,682	0.35	2,657,490	0.04	14	81.76
chaos	12,645	0.11	1,583,944	0.02	2,887	143.26
rarp	18	0.00	840	0.00	9	64.67
other (ip)	25	0.00	1,228	0.00	0	67.12
other	8,386	0.07	473,388	0.00	6,923	74.45
Total	11,837,073	100.00	6,847,096,334	100.00	156,849	596.45

The table is somewhat misleading in that it lists protocols that reside at different layers of the ISO OSI architecture. For instance, NFS, which uses UDP as its transport protocol, covers portions of the *transport* and *session* layers. Since we have classified NFS independently, the figures for UDP in Table 4-1 correspond to the component of UDP that is not used by the NFS protocol. We have chosen to divide the traffic according to protocol's functions rather than according to the ISO model in part because it is difficult to place each protocol at one particular layer, in part because our classification serves as a basis for the rest of the analysis in this study.

One of the reasons for which the disk and file access protocols generate such a large portion of the traffic is that the physical memories in our workstations are rather small– ranging from 2M to 4M bytes (see Table 2-1)– in comparison to the demands of the software run today. In particular, as more protocols are added to it, the size of the kernel increases, graphic libraries are very large, and common programs' working sets grow as software becomes more sophisticated. The activity of the ND protocol is particularly dominant. As we shall see in Section 6, most of this activity is *system* activity, which includes paging and swapping, accounting, and the like. This observation questions the common belief that the bulk of system operation is directly related to the user-generated workload; rather, it appears to be caused by the system's own managerial tasks unrelated to user activity. We shall deal with these issues in more detail in Sections 6 and 7.

Both ND and NFS, as well as TCP, UDP, and ICMP, use the protocol IP as the underlying network layer. The IP protocol alone is responsible for 97.76 percent of the total packet count and for 99.62 percent of the total byte count. In the next two sub-sections, we analyze the traffic communication patterns of IP packets for two reasons. First, it is very convenient because an IP packet header provides both the network and host numbers for the source and destination addresses. Second, IP traffic accounts for virtually all of the traffic.

Table 4-1 shows also the number of broadcast packets for each protocol. A packet is classified as a broadcast packet if all bits in the destination address of the Ethernet packet header– a 48-bit quantity in a 10 Mb/s Ethernet– are set to 1. There are relatively few broadcast packets, 1.33 percent of the total. Of the total number of broadcast packets only 10.96 percent are IP broadcast packets; however, they transport one half of the bytes that are broadcast.

The Address Resolution Protocol [43] (ARP), which is used to convert protocol addresses (most often IP addresses) to 48-bit Ethernet addresses, accounts for 69.4 percent of the broadcast packets. A station that does not find the Ethernet address in a locally maintained cache will broadcast a request packet containing the protocol address to all hosts in the network. The host that identifies the protocol address as its own address will reply (a non-broadcast packet) to the inquiring station. In our network 94.9 percent of the ARP packets are request packets with no corresponding responses. We discovered that in most cases they are sent by machines that do not understand subnetwork addresses. These machines (the LISP machines on our network) generate broadcast IP addresses that are not understood by the other hosts and therefore never receive a response ARP packet. If these problems, which prevent ARP host caches from functioning, were corrected, the percentage of broadcast packets would significantly decrease. From the data and these observations, we conclude that the broadcast mechanism exerts a negligible influence in an Ethernet. Researchers have used this fact to support two antithetical conclusions: for some there is room to expand the usage of the broadcasting mechanism;[12,25] for others, the mechanism is unnecessary and its use should be discouraged.[47]

4.4. Traffic Patterns

Table 4-2, constructed from the source and destination fields of the IP packet headers in the traces, helps to understand the source-destination traffic patterns. SUN file servers are in boldface, VAX's in capital letters, and MicroVAX's and LISP machines in italics.

The table is divided into two parts. Each line on the left side lists the traffic between a source and a destination machine as a percent of the total IP traffic in the traces. The right side lists the traffic between the same two machines in the opposite direction. Vertically, lines are arranged in decreasing order of total number of bytes transmitted (the sum of the values in columns four and eight); horizontally, within one line, the entry with the higher number of bytes transmitted has been placed on the left. Notice that the entries of Table 4-2 comprise 79.2 percent of the number of IP packets and 96.5 percent of the number of IP bytes, or 77.4 percent of the total number of packets and 96.1 percent of the total number of bytes.

The traffic is highly unbalanced. The two machines at the top of the table are responsible, alone, for 19.6 percent of the number of packets and 27.4 percent of the total number of bytes. The top three lines comprise almost 41 percent of the total byte count. This fact should be considered carefully by system administrators, who often make poor assumptions about the traffic patterns under which a local area network will operate. It would be, for instance, a bad idea to place two machines that generate a high number of packets at the opposite ends of an Ethernet cable as this will increase the time window for detecting collisions. In particular, the logical position for file servers would be the center of the cable near to each other [one would have to modify this statement in the case of multi-cable Ethernets].

SUN workstations dominate the traffic as they exchange among themselves almost 95 percent of the total byte count; VAX-to-VAX traffic contributes only two percent of the total; and SUN-to-VAX traffic accounts for another two percent. Notice that the Xerox Star workstations use the Xerox NS family of protocols and therefore are not listed in Table 4-2. However, as one can infer from the statistics in Table 4-1, at Berkeley, in that period, they were mostly used as stand-alone systems.

From the *PACKETS* and *BYTES* columns we can obtain a rough idea of the traffic *protocol mix* for each pair of machines. For the machines at the top of the table, ND and NFS account for most of the packets exchanged; for machines at the bottom of the table, TCP accounts for most of the packets exchanged.

4.5. Intra-network Traffic and Inter-network Traffic

By looking at the addresses in the IP packet headers, we can divide the traffic into three categories: 1) traffic that remains within XCS (local traffic); 2) traffic that goes outside of XCS but remains within the UCB boundaries; and 3) traffic directed to or generated by machines outside the Berkeley local area networks. (Occasionally, we see a few packets with wrong IP addresses; for instance, some of the MicroVAX's generated several packets with IP destination addresses that

TABLE 4-2. SOURCE-DESTINATION TRAFFIC STATISTICS (VALUES IN PERCENT)

SOURCE	DESTINATION	PACKETS	BYTES	SOURCE	DESTINATION	PACKETS	BYTES
sequoia	madrone	10.6721	16.2099	madrone	sequoia	8.9428	11.2363
ginko	teak	2.6637	3.7717	teak	ginko	2.3927	3.3442
snow	bashful	2.4901	3.8745	bashful	snow	2.1357	2.4291
xcssun	saturn	1.5745	2.4607	saturn	xcssun	1.3454	1.5245
xcssun	jupiter	1.4763	2.2784	jupiter	xcssun	1.2436	1.4227
xcssun	mars	1.3162	1.9891	mars	xcssun	1.1699	1.3374
sequoia	liveoak	1.1917	1.6308	liveoak	sequoia	1.1292	1.3437
snow	doc	1.1155	1.7239	doc	snow	0.9725	1.1336
xcssun	uranus	1.0317	1.5645	uranus	xcssun	0.9351	1.1289
xcssun	venus	1.0351	1.5800	venus	xcssun	0.8784	0.9903
sequoia	redwood	0.8715	1.3349	redwood	sequoia	0.7375	0.8908
dim	chip	0.9121	1.4677	chip	dim	0.8784	0.6947
xcssun	neptune	0.7684	1.1713	neptune	xcssun	0.6804	0.7934
snow	happy	0.7626	1.1218	happy	snow	0.7182	0.8402
xcssun	mercury	0.7328	1.1022	mercury	xcssun	0.6417	0.7301
teak	eucalyptus	0.6399	0.9453	eucalyptus	teak	0.5373	0.6406
snow	grumpy	0.6496	1.0380	grumpy	snow	0.4696	0.4780
sequoia	elm	0.5641	0.7050	elm	sequoia	0.6008	0.7007
teak	bansai	0.4867	0.6830	bansai	teak	0.4854	0.6467
pine	lassen	0.5269	0.8095	lassen	pine	0.3811	0.4871
sequoia	fir	0.4943	0.6688	fir	sequoia	0.5594	0.6202
ash	sequoia	0.5800	0.7641	sequoia	ash	0.4073	0.5100
baobab	lassen	0.5219	0.7863	lassen	baobab	0.3819	0.4859
rip	lassen	0.5401	0.8877	lassen	rip	0.2803	0.2748
walnut	sequoia	0.4802	0.7739	sequoia	walnut	0.2535	0.2370
maple	lassen	0.4319	0.7130	lassen	maple	0.2156	0.1967
pawlonia	lassen	0.4217	0.6911	lassen	pawlonia	0.2155	0.1996
geppetto	teak	0.3389	0.5005	teak	geppetto	0.2716	0.3510
palm	teak	0.3700	0.5991	teak	palm	0.2049	0.2044
mike	dim	0.2861	0.4012	dim	mike	0.2163	0.2694
lassen	ginko	0.4155	0.3432	ginko	lassen	0.4002	0.2928
larch	teak	0.3247	0.5775	teak	larch	0.0966	0.0330
lassen	sleepy	0.2227	0.3412	sleepy	lassen	0.1944	0.1694
snow	pinocchio	0.2291	0.3747	pinocchio	snow	0.1286	0.1041
snow	sneezy	0.2203	0.3487	sneezy	snow	0.1360	0.1286
snow	dopey	0.2076	0.3255	dopey	snow	0.1303	0.1239
dim	rob	0.1863	0.2776	rob	dim	0.1374	0.1450
sequoia	shangri-la	0.1912	0.3001	shangri-la	sequoia	0.1133	0.0282
sequoia	DALI	0.1783	0.2857	DALI	sequoia	0.0892	0.0070
allspice	KIM	0.1690	0.2711	KIM	allspice	0.0856	0.0079
envy-150	KIM	0.1330	0.2159	KIM	envy-150	0.0672	0.0056
ginger	MIRO	0.1116	0.1822	MIRO	ginger	0.0605	0.0047
ginko	INGRES	0.7641	0.0709	INGRES	ginko	0.5332	0.0543
JI	ERNIE	0.5104	0.0550	ERNIE	JI	0.6032	0.0516
shangri-la	INGRES	0.2444	0.0853	INGRES	shangri-la	0.0975	0.0095
uranus	VANGOGH	0.2112	0.0625	VANGOGH	uranus	0.1623	0.0294
CORY	*dikdik*	0.2268	0.0494	*dikdik*	CORY	0.2661	0.0276
xcssun	pluto	0.0386	0.0643	pluto	xcssun	0.0199	0.0095
sequoia	ginko	0.0549	0.0515	ginko	sequoia	0.0432	0.0136
JI	CORY	0.2499	0.0344	CORY	JI	0.3598	0.0283
RENOIR	elm	0.1817	0.0395	elm	RENOIR	0.2191	0.0190
RENOIR	ash	0.2743	0.0286	ash	RENOIR	0.3197	0.0272
CORY	*plato*	0.0514	0.0506	*plato*	CORY	0.0356	0.0048
achilles	KIM	0.3296	0.0258	KIM	*achilles*	0.2363	0.0225
chip	ERNIE	0.3594	0.0281	ERNIE	chip	0.2126	0.0193
RENOIR	redwood	0.1955	0.0284	redwood	RENOIR	0.2254	0.0176
UCBARPA	JI	0.2773	0.0222	JI	UCBARPA	0.1884	0.0203
RENOIR	fir	0.1870	0.0235	fir	RENOIR	0.2219	0.0174
CORY	*aristotle*	0.2255	0.0212	*aristotle*	CORY	0.1650	0.0176
neptune	DOROTHY	0.3489	0.0273	DOROTHY	neptune	0.1018	0.0113

had the high-order 16 bits equal to zero. These anomalies are part of a more general class of problems related to the generation of return IP addresses and will be discussed in Section 5.)

All of the file system traffic (either carried by the ND or NFS protocol) is local. Although it would be possible to place client workstations and their file servers or ND servers on different networks, given the large amount of data that these protocols carry, gateways would be a potential problem. Should they run out of buffer space during the periods of peak utilization, they would drop packets, increasing the protocols' response times. UDP traffic, most of which is generated by the routing daemons, is also almost entirely local. Inter-network traffic consists of character and file transfer traffic, which is transported by the TCP protocol. In the remainder of this sub-section all data is expressed as percentage of the total number of TCP packets and total number of TCP bytes.

TABLE 4-3. SOURCES AND DESTINATIONS OF TCP TRAFFIC (PERCENTAGES)

		DESTINATION			
		XCS	ucb	outside	Total
SOURCE	XCS	10.37 17.33	40.66 33.12	0.37 0.16	51.40 50.61
	ucb	32.77 23.58	15.58 25.62	0.00 0.00	48.35 49.20
	outside	0.25 0.19	0.00 0.00	0.00 0.00	0.25 0.19
	Total	43.39 41.10	56.24 58.74	0.37 0.16	100.00 100.00

Table 4-3 divides TCP traffic according to the source and destination networks of IP addresses: we label with *xcs* the traffic that was directed to or coming from hosts in the XCS network; with *ucb* all the traffic, except for the XCS traffic, that remains within the UCB networks boundaries; and with *outside* the traffic that originates or is directed outside UCB. For each combination of source and destination addresses, we show two figures: on the top, the percentage of the total packet count and, on the bottom, the percentage of the total byte count. First we see that only 10.37 percent of the total TCP packet traffic is local; 89.01 percent is inter-network traffic within UCB; and the remaining 0.62 percent is inter-network traffic going to or coming from non-UCB networks. The table also shows that approximately half of the total TCP traffic is generated by hosts on XCS and half by hosts on other UCB networks. Only a negligible portion is generated by hosts outside UCB. The reason for these numbers is that file transfers across wide area networks between distant machines are rare. Most traffic to distant machines is generated by electronic-mail applications, and at Berkeley, as we have seen, users read and send their mail on machines that do not reside on the XCS network.

Table 4-4 breaks down the local and UCB inter-network TCP traffic. Blanks in the table indicate that no traffic between the given endpoints passes through XCS. Section 2 describes the listed networks and Figure 2-1 shows a partial topology of the Berkeley domain. Both ucb-ether, a 2.94-Mb/s Ethernet, and the spine subnet connect the most important VAX's used by Computer Science division users. This is reflected in the high proportion of traffic on XCS to and from these two networks. The table indicates that more traffic flows from XCS to ucb-ether rather than in the opposite direction (21.28 vs. 11.15 percent of the total byte count). One of the reasons for this is that some SUN file systems were not, at the time of the measurements, backed up and users often explicitly copied important data on VAX's file systems.

A good deal of traffic on XCS originates on spur, another network of SUN workstations. Since spur's only gateway, *spur-gw*, is on XCS, spur's traffic must go through XCS in order to reach other networks. However, because the two networks are used by different research groups, with different sponsors, traffic between XCS and spur is low.

We call the traffic that originates on machines outside XCS and is directed to machines

TABLE 4-4. SOURCE AND DESTINATION NETWORKS OF TCP TRAFFIC WITHIN BERKELEY CAMPUS

		DESTINATION								
		XCS	ucb-ether	cs-div	cad	eecs	cc	spine	spur	other
SOURCE	XCS	10.37 17.33	22.16 21.28	3.14 3.64	2.21 1.12	4.50 2.08	0.40 0.18	7.77 4.18	0.42 0.50	0.07 0.14
	ucb-ether	17.45 11.15							4.51 2.66	
	cs-div	2.54 2.81							0.06 0.07	
	cad	1.59 0.99							0.87 0.50	
	eecs	2.92 3.20								
	cc	0.35 0.17							0.04 0.02	
	spine	7.39 5.03							1.12 0.80	
	spur	0.48 0.21	6.20 17.80	0.06 0.03	1.26 3.08		0.05 0.02	1.40 0.64		
	other	0.04 0.02								

outside XCS *traffic in transit*. This traffic is represented in Table 4-3; it accounts for 15.58 percent of the packets and 25.62 percent of the bytes. Table 4-4 shows that the spur network is almost exclusively responsible for the traffic in transit.

TABLE 4-5. TCP TRAFFIC SWITCHED BY XCS GATEWAYS (PERCENTAGES)

GATEWAY	DATA TRANSMITTED		DATA RECEIVED	
	PACKETS	BYTES	PACKETS	BYTES
dali	28.02	19.87	23.84	19.13
calder	12.19	8.99	25.56	37.04
spur-gw	9.71	22.78	8.26	5.22
Total	49.92	51.64	57.66	61.39

Table 4-5 shows the percentage of total TCP traffic that the three gateways on XCS switched. The three machines together switched more than 400 million TCP bytes, or 100 million more than the total traffic of Shoch and Hupp, and Calder alone routed 164M bytes, more than 50 percent of the Xerox PARC total. This is possible (recall that the total TCP byte count in Table 4-1 amounts to 356.6M bytes) because a single TCP packet in transit is counted twice: as an incoming packet on some gateway, and as an outgoing one on another. We observed 540,876 packets in transit, which transported more than 95M bytes.

Table 4-6 displays how this transit traffic is divided among the gateways. As we have seen, the bulk of the traffic in transit is due to TCP traffic created by the spur network. Notice a curious phenomenon: most of the communication from spur to the VAX's is switched by calder to ucb-ether (see Figure 2-1). However, most of the return traffic is switched by dali from the cs-div network (recall that most of the Computer Science VAX's are on both ucb-ether and cs-div).

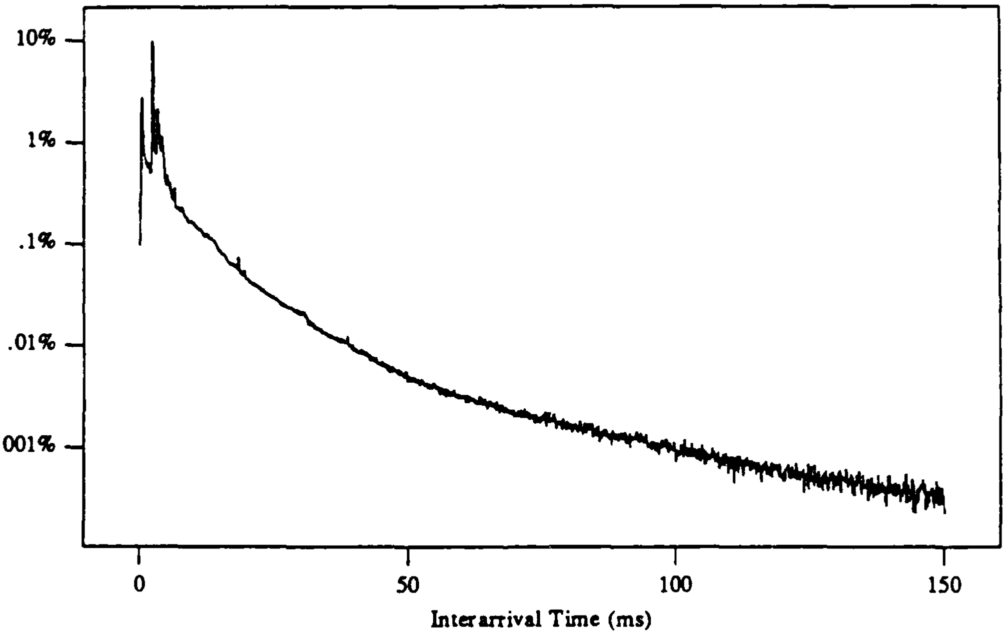
4.6. Interarrival Time

Figure 4-6 illustrates the distribution of packet interarrival times over one day. We compute the interarrival time as the difference between the times when the transmissions of two subsequent packets began. Note that due to the high arrival rate, there are a small number of interarrival times greater than 100 ms. By contrast, in measurements taken at MIT[19] there is a significant percentage of arrivals with interarrival times above 200 ms.

TABLE 4-6. DISTRIBUTION OF TRANSIT TRAFFIC AMONG GATEWAYS

		DESTINATION			
		calder	dali	spur-gw	Total
SOURCE	calder		0.00 0.00	9.44 3.71	9.44 3.71
	dali	0.03 0.01		32.95 12.10	32.98 12.11
	spur-gw	40.41 78.41	17.17 5.77		57.58 84.18
	Total	40.44 78.42	17.17 5.77	42.39 15.81	100.00 100.00

FIGURE 4-6. PERCENTAGE OF PACKET ARRIVALS (ALL PACKETS)

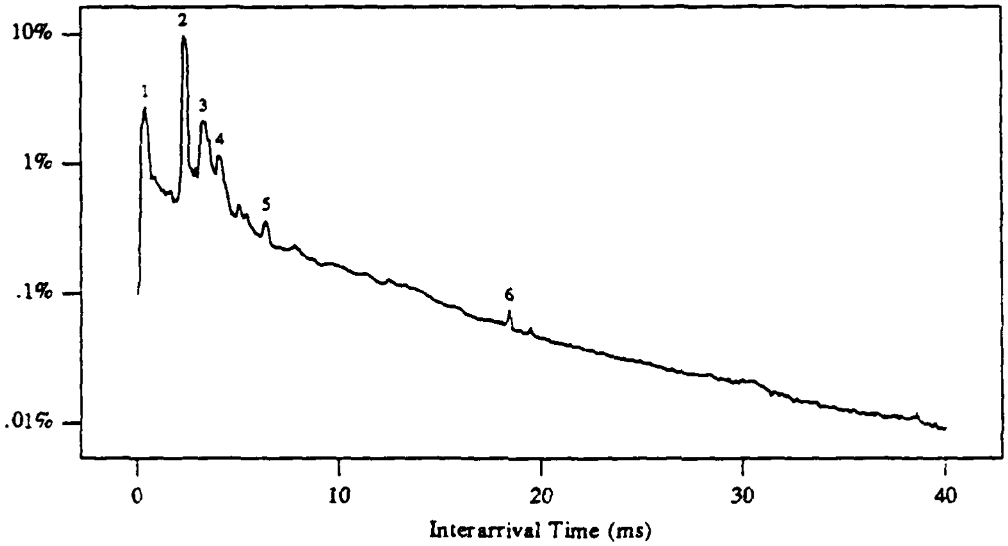


Analytical or simulation studies often model the arrival process as a Poisson process or as a batch Poisson process (sometimes called a compound Poisson process). [The absence of correlation between arrivals, which is one key assumption of the Poisson model, is essential to obtain closed-form analytic solutions.[6]] In the first case, a semi-logarithmic scale graph of the interarrival time, like the one represented in Figure 4-6, would be a straight line. In the second, it would be a straight line with a peak– the batch– at the origin.

The Poisson model is inappropriate because it assumes that there is no correlation between arrivals. Preliminary investigation shows that the arrival process is highly bursty and that the independence assumption is almost certainly not justified.[20] Figure 4-6 shows that there is a high probability that one arrival will be followed by a second one within a deterministic time, which depends or the protocol, the packet sizes, and the traffic intensity. Three facts explain this: first, the objects transmitted by network protocols are much larger that the maximum packet size on the Ethernet; second, request-response protocols (both ND and NFS can be so classified) generate interarrival times whose distributions have pronounced modes; and third, the sources of packet traffic are bursty.[26, 53] Another reason for rejecting the Poisson model as well as the batch Poisson one is that the mean of our arrival process is not constant (see Figure 4-9). Furthermore, in

Figure 4-7, where we show the detail of the interarrival times in the region from 0 to 40 ms, we see that there are a number of peaks located between 0 and 7 milliseconds. In this paper, by studying the various protocol components of the traffic separately (see Sections 5.2.5, 6.4, and 7.7), we show that we can characterize interarrival time constants that are not visible if the arrival process is studied globally. However, within the limited scope of this article, we do not try to answer the difficult and challenging question of what the most appropriate model for a network of diskless workstations is. A related paper will deal with these issues in greater detail.[24]

FIGURE 4-7. PERCENTAGE OF PACKET ARRIVALS (ALL PACKETS) DETAIL



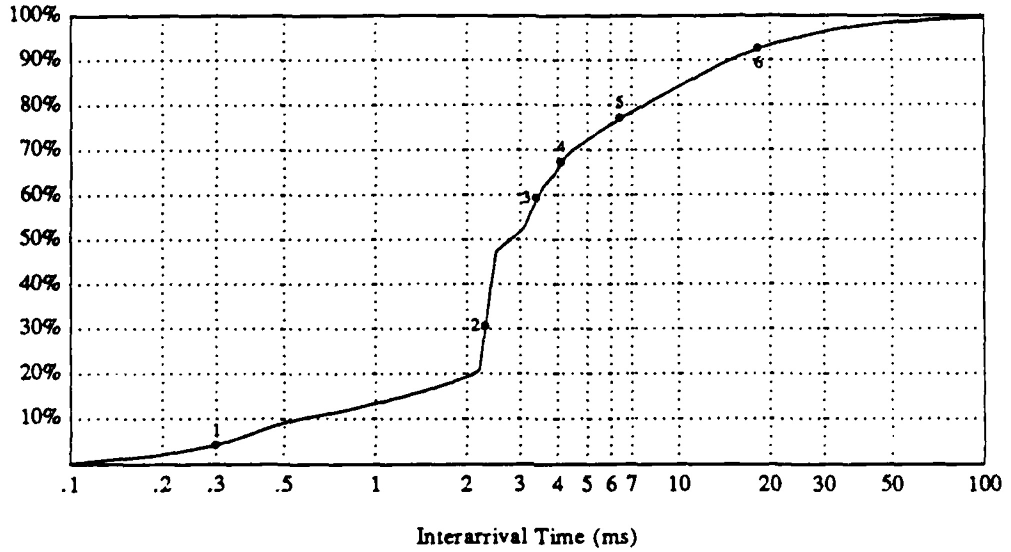
We have numbered the peaks in Figure 4-7; as observed above, they represent patterns of interarrival time. Each of these peaks, which is distinct from each other, has a width of no more than a third of a millisecond. This is in accord with the analysis of the timing errors in our measurements that we performed in Section 3 All of these peaks are generated by the ND protocol and will be explained in Section 6.4. They correspond, in order, to peaks 1, 3, 5, 6, 8, and 12 in Figure 6-6.

Figure 4-8 displays the cumulative distribution of the interarrival times. The numbers refer to the peaks in Figure 4-7. Notice that 50 percent of the packets are followed within 3 ms by the next packet, 84 percent are followed within 10 ms by another, and 99 percent within 90 milliseconds. Shoch and Hupp observed values three times as large.

There are several factors that contribute to this variation. First, the network utilization is higher, which makes the mean interarrival time lower. The mean packet arrival rate is 137 packets per second and the mean interarrival time is 7.299 ms, while the corresponding figures for the Xerox PARC Ethernet were 25 packets per second and 39.5 ms respectively. Second, the bit rate in our Ethernet is 10 Mb/s while it was 2.94 Mb/s in the case of the older Ethernet. Third, the newer protocols, in particular the ND protocol, have been optimized for the shortest response time. Finally, our SUN's and MicroVAX's are faster than the Alto computers on the Xerox Ethernet and many of our Ethernet interfaces [SUN Intel and AMD Am7990 Lance] provide much higher performance than the Experimental Ethernet controller† in exclusive use at Xerox PARC eight years ago.

† The Experimental Ethernet controller for the Alto was half-duplex. The time required to switch to receiving mode after packet transmission could have caused packets to be missed by the receiver.[52]

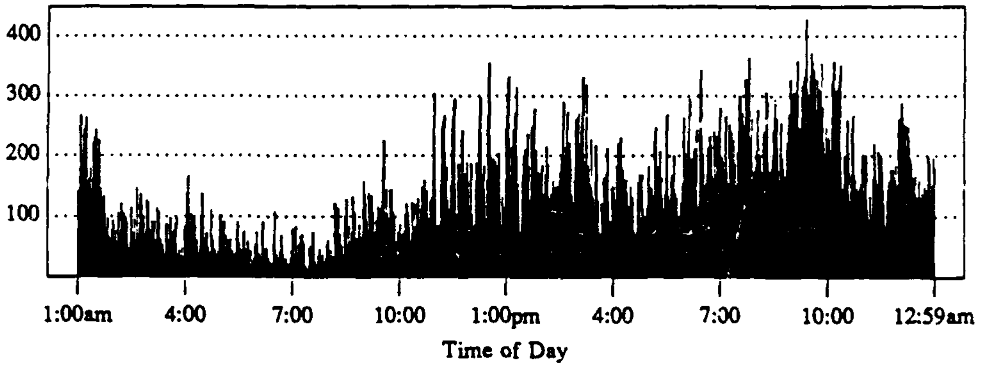
FIGURE 4-8. CUMULATIVE PERCENTAGE OF PACKET ARRIVALS (ALL PACKETS)



4.7. Packet Rate

Figure 4.9 displays the packet arrival rate averaged over one-minute intervals. It shows that the rate is a function that varies with the time of day. Observe that, despite the non-uniform packet size distribution (Figures 4-4 and 4-5), the packet arrival rate is a very good estimator of the network utilization (Figure 4-1).

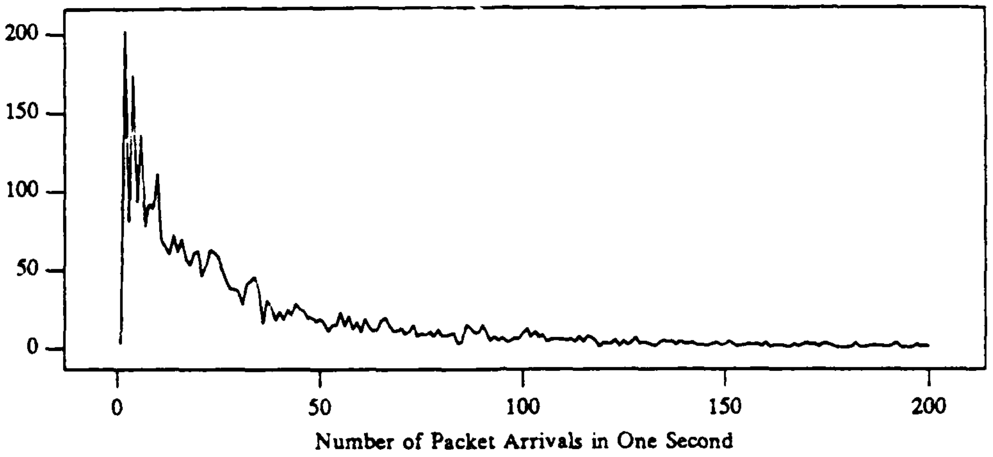
FIGURE 4-9. PACKET ARRIVAL RATE (PACKETS/SEC)



In order to obtain a better understanding of the variation in the packet arrival rate, we counted how many packets arrived over one-second intervals and then counted how many of those one-second intervals had the same number of packet arrivals. Figure 4-10 and Figure 4-11 plot these numbers for two time periods: between 7 and 9 am, at low network utilization; and between 9 and 10 pm, when network utilization is at its highest. The highest number of packets received in one second was 343 in Figure 4-10 and 572 in Figure 4-11; in both cases we have truncated the the graphs for the sake of graphical presentation.

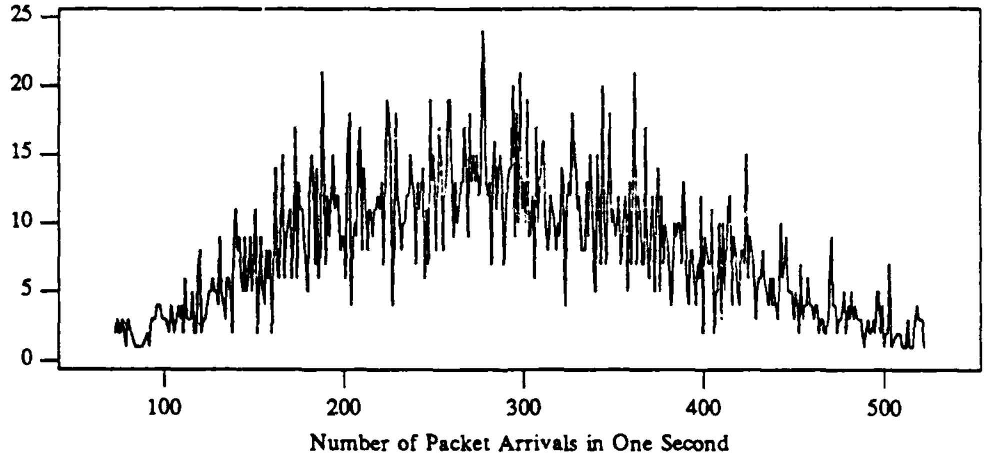
The first three peaks in Figure 4-10 occur at packet arrival rates of 2, 4, and 6 packets per second. Odd-numbered arrival rates are less frequent because our protocols tend to reply to each request packet. In the case of TCP there are transmission/acknowledgement pairs, while for ND and NFS there are request/response pairs. The oscillations in Figure 4-11 are instead due to the small number of samples available for each sampling point.

FIGURE 4-10. NUMBER OF INTERVALS BETWEEN 7 AND 9 AM



Given that the sampling interval of one second is larger than the protocol time constants, which are in the order of tens of milliseconds, the number of arrivals in one second can be thought of being independent of the arrivals in the following second. Under this condition, the number of arrivals in one-second intervals could be modeled by a discrete Poisson distribution. The shape of the two figures confirms this hypothesis.

FIGURE 4-11. NUMBER OF INTERVALS BETWEEN 9 AND 10 PM



4.8. Summary

Our measurements show that packet traffic in a diskless workstation environment differs sharply from the traffic in other environments.

First, we noticed how the network utilization has increased due to the new file system and disk protocols. For significantly long periods during the busiest hours of the day, Ethernet utilization amounts to about a third of its raw total bandwidth. Studies have shown that, at this level, network latencies will begin to increase.[30] For instance, according to Gonsalves,[22] who measured Ethernet delays at varying, artificially-generated network loads for different packet lengths, the queuing time at traffic intensity 0.3 is larger than 5 ms when all packets transmitted are 64 bytes long, and about 19 ms when the packet length is 1500 bytes.† We shall see in Section 7 how the increased latency affects the NFS protocol.

Second, the communication patterns are skewed. Although in the Xerox Ethernet more bytes were directed to network servers, the percentage of traffic between any two single machines was small compared with the total traffic.†† Not only have we seen that two single machines can load our Ethernet to one fifth of the raw network bandwidth, but they can also generate 20 percent of the total packet count. This phenomenon should be attributed both to differences in users' behavior and to differences in hardware efficiency. In particular, low performance network interfaces (such as the 3Com Ethernet interface which only has buffer space for two packets and does not allow DMA to the host's memory) will not achieve high throughput. These observations strongly suggest that, under our traffic characteristics, the Ethernet may well become a performance bottleneck.

The fact that terminal and file transfer traffic carries more data in our network than it did in the Xerox network should be attributed to two related factors. First, our workstations have larger bitmap displays and a very flexible user interface; as a result, users keep several windows open during one session and log on to a number of machines. Second, the topology of the network environment within a single organization most often involves more than a single Ethernet. As a result, often a network carries transit traffic that is originated from and destined to machines not connected to it. A portion of the file transfers are due to the fact that the SUN file systems were not backed up. The question arises whether local disks (used for the file system rather than as backing store) would have modified the network traffic due to file copying. We do not believe, however, that the local disks of the Alto workstations used at Xerox PARC modified significantly the proportion of the traffic due to remote file copy.

Third, the median of the packet size distribution has increased because of the large amount of file page traffic. The distribution itself is no longer bimodal.

Fourth, request-response protocols associated with fast network interfaces have reduced protocols' time constants. The arrival process shows a significant departure from a Poisson process.

Although these general observations offer many insights, we can gain further knowledge by analyzing the protocols that are responsible for most of the traffic. In the following three sections we shall describe the network protocols by focusing on TCP, ND, and NFS. We shall limit the analysis to those features of the protocols that are relevant to understanding communication traffic. Thus, the file access properties of ND and NFS will not be covered in detail.

5. The DARPA Family of Protocols

The DARPA family of protocols spans several layers of the Open System Interconnection (OSI) model developed by the International Standards Organization (ISO). At the *application* layer, it includes a virtual terminal protocol (TELNET), a file transfer protocol (FTP), and a mail transfer protocol (SMTP). In Berkeley UNIX 4.2 the first two are implemented as user programs, *telnet* and *ftp*, while the latter is implemented by the mailing daemon *sendmail*. At the highest layer, Berkeley UNIX also provides a remote login protocol (*rlogin*, *rlogind*), another remote file copy protocol (*rcp*), and a remote execution protocol (*rexec*). Together they provide a comprehensive set of network services over both local and wide area networks. All of these protocols are built on the communication mechanisms provided by the Transmission Control Protocol, a session and transport layers protocol, and the Internet Protocol, a network layer protocol.

† The transmission time of a 1500-byte packet is roughly 1.2 ms.

†† We infer this from Figure 5 of Shoch and Hupp's paper, as the authors do not present quantitative data for their communication matrix.

5.1. The Internet Protocol and the Internet Message Control Protocol

The Internet Protocol was designed for the Arpanet, a packet switched network. The protocol performs two major functions: addressing and fragmentation. Since packets, which in the Arpanet terminology are called *internet datagrams*, can take one of several routes, each packet header indicates the source and destination addresses. The protocol treats each datagram as an independent entity unrelated to any other internet datagram. At this level, there are no connections or virtual circuits. In the process of being forwarded to its destination, a large packet could be routed to a network that only allows smaller packet sizes. The IP modules are then responsible for the fragmentation and reassembly of datagrams.

There are no mechanisms to improve the end-to-end data reliability. There is no flow control: if a network becomes congested the IP modules may choose to drop incoming datagrams. There is no sequencing and packets may arrive out of order. There is no error control; only the IP header is checksummed.

An internet module resides on each host engaged in communication and in each gateway. Occasionally a gateway or a destination host will communicate with a source host, for example, to report an error in datagram processing. For such purposes they use the Internet Control Message Protocol (ICMP).[44] ICMP uses the basic support of IP as if it were a higher level protocol; however, ICMP is actually an integral part of IP, and must be implemented by every IP module. ICMP messages are sent in several situations: for example, when a datagram cannot reach its destination, when the gateway does not have the buffering capacity to forward a datagram, and when the gateway can direct the host to transmit traffic on a shorter route.

The Internet Protocol is a host-to-host protocol in that there is no provision for addressing a particular process running on a host. The IP header contains a type field, which is used at the destination hosts to demultiplex the data carried by internet datagrams to higher-level protocols. The most common of these protocols are the Transmission Control Protocol and the User Datagram Protocol (UDP). The modules associated with these protocols are responsible for addressing processes within each host. (Notice that SUN Microsystems has defined a non-standard, new IP type for its Network Disk protocol, which uses IP as its transport protocol. Since ND carries disk blocks between a machine's kernel and remote disk servers, a host-to-host protocol is appropriate.)

It is important to notice that IP addresses, 32-bit quantities, are conventionally divided into segments that represent network and host numbers respectively. While there are several different representations,[21, 28, 39, 40, 46] the one most often used at Berkeley, which allows the logical division of a local area network into subnetworks, divides the IP address into four 8-bit quantities: an internet network number, a local network number, a subnetwork number, and a host number. There are provisions in IP for transmitting broadcast messages if a network allows it. Although there are proposals for an extension,[18] currently network gateways do not forward broadcast messages to networks other than the one where the message originated.

We noticed that a number of packets with erroneous IP destination addresses were transmitted. In some instances, we saw ICMP packets whose source address was the IP broadcast address.† Some IP packets were broadcast although their IP destination address was not the broadcast address. We observed packets with the address of the software loopback interface, which is used by a machine when a packet is sent to its own address, as source address. A few IP packets whose destination was the broadcast address were not broadcast on the Ethernet.

All these anomalies have very little statistical significance, but packets with wrong addresses can cause a good deal of trouble. Whenever an IP packet with a wrong address is received by a machine, the IP module on that machines returns to the sender an ICMP error message. If the faulty packet is broadcast, every receiving station will attempt to reply simultaneously causing

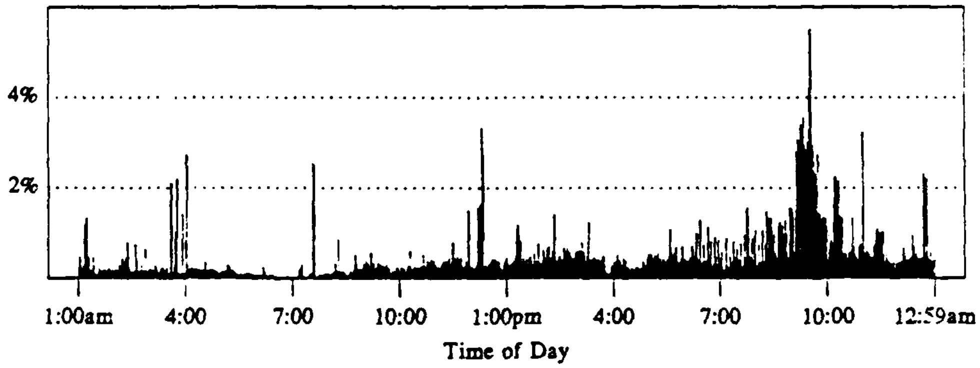
† Some implementations of TCP/IP simply reverse the source and destination addresses of an ICMP *echo* message when constructing the *echo reply* to send to the requesting machine. When the destination address of the *echo* message is the broadcast address, Berkeley UNIX instead substitutes the address of the network interface that received the *echo* ICMP message for the IP source address of the *echo reply*.

network congestion. In view of these problems, which are particularly severe on gateways, at Berkeley system administrators are considering the establishment of requirements that a host must satisfy in order to be connected to a network.[29]

Of all ICMP packets received, 63.43 percent were *redirect* messages for the network. A gateway sends a redirect message to a host that wants to communicate with a second machine when the gateway's routing table shows that the IP message should be sent to another gateway that is on the host's network. The redirect message advises the host to send its traffic for the second machine directly to the gateway on its network as this is a shorter path to the destination. As many as 8.87 percent of the ICMP messages were *destination unreachable* messages and only 0.13 percent were *source quench* message. Since the latter are sent by hosts or by gateways when their receiving capacity limit has been reached (or exceeded) to inform the sources that they should reduce the transmission rate, we conclude that there is adequate buffer space on the hosts for today's IP traffic at the throughput levels of SUN UNIX and Berkeley UNIX 4.3BSD.

As shown in Table 4-1, the IP protocol accounts for the large majority of packets transported by our Ethernet. The three most important components are those of TCP, a traditional connection-oriented protocol; ND, a specialized request-response protocol; and NFS, which is based on a remote procedure call protocol. In the remainder of Section 5 we shall describe the traffic characteristics of TCP and UDP; and in Sections 6 and 7, those of ND and NFS.

FIGURE 5-1. ETHERNET UTILIZATION (TCP PROTOCOL)



5.2. The Transmission Control Protocol

TCP is a connection-oriented protocol that provides reliable process-to-process communication service between two machines in a multi-network environmert. Reliability is achieved by using sequence numbers for the data transmitted and requiring positive acknowledgement from the receiving side. Unacknowledged data is retransmitted after a timeout interval. A receiving TCP module uses the sequence numbers to correctly order packets received out of order and to eliminate possible duplicates. Damaged packets, which are discarded, are detected by verifying that the received data generates a checksum value equal to the one added to each transmitted packet. TCP provides flow control by returning with every acknowledgement a *window* that indicates the number of bytes for which the receiver has allocated buffer space.

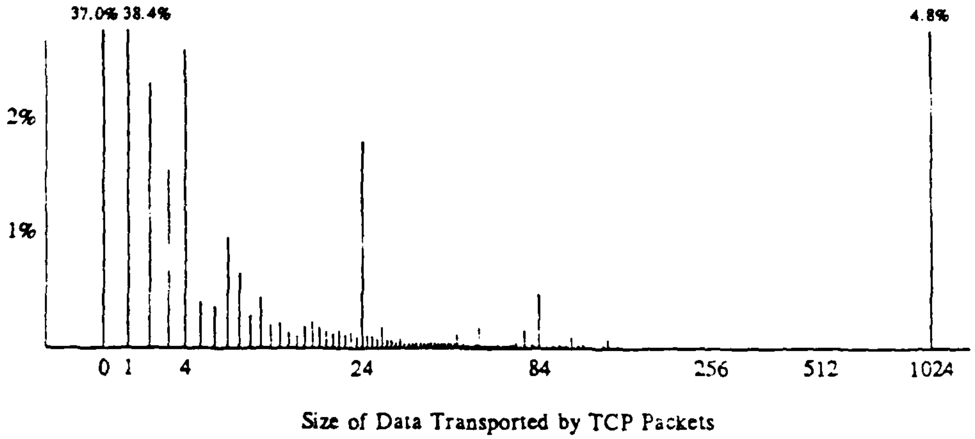
5.2.1. Utilization

In Figure 5-1 we show the network utilization due to the TCP protocol. The most striking aspect of this protocol is that, despite the high number of packets transmitted, it generates very low network utilization. The activity peak between 9 and 10 pm is due to file system dumps performed across the network. Notice that the file system dumps may not be good indicators of TCP throughput since, in the presence of disk contention, data may not be generated so fast as to keep TCP buffers always full. In addition, the programs used for the dumps may be inefficient.

5.2.2. Data Size

Many TCP packets carry character traffic (characters of UNIX shell command lines) from diskless workstations on the XCS network to VAX's on other networks. Therefore, many packets are short. This is displayed in Figure 4-4, which also shows that TCP traffic is essentially bimodal. Figure 5-2 shows the size of the objects transported by TCP packets, i.e. the length of the data portion of TCP packets. More than a third of them transport no data. They are acknowledgement packets (each carrying a 20-byte IP header, a 20-byte TCP header,† and 6 empty data bytes!) that are sent from the receiving host to the transmitting one. Although a portion of the one-byte objects is due to editors that put tty devices in *raw mode* and do not buffer character transmission, most of them are *keep alive* messages. Keep alives, which involve two messages, are implemented by periodically retransmitting the last byte of a conversation (a one-byte message) in order to get an acknowledgement (a zero-byte message) from the other side of a connection.††

FIGURE 5-2. PERCENTAGE OF TCP PACKETS VS. DATA SIZES



Notice that the longest object is 1024 bytes long. TCP throughput would increase if, when needed, maximum size packets were used.

The two peaks at 24 and 84 bytes depend almost exclusively on packets transmitted by connections established by *xterm* in the X window system,[50] which. at the time of the measurements. was used on XCS's MicroVAX's. Each 24-byte message carries a single byte of information encapsulated in a structure used for the management of graphical representation. It is the traffic of 24-byte data messages that generates the small peak at 64 bytes in the global packet size distribution in Figure 4-4.

Berkeley UNIX 4.2, as well as 4.3BSD, but not SUN UNIX, employs the so-called *trailer* encapsulation.[33] In this encapsulation, when a message to be transmitted is sufficiently long, the protocol information is placed at the end of the packet so that the received data can be page-aligned and copy operations can be reduced to manipulation of page pointers. The number of TCP packets with trailer encapsulation transmitted in one day is 18,826.

In the past, when TCP was mainly used on time-sharing machines with many active processes, there was concern about the kernel memory to allocate for transmission buffers and status information. Today, with single-user machines, and decreasing memory costs, the the window size of TCP has increased. In our data, window sizes of 2K and 4K bytes were most common and some lisp machines used windows as large as 16K bytes.

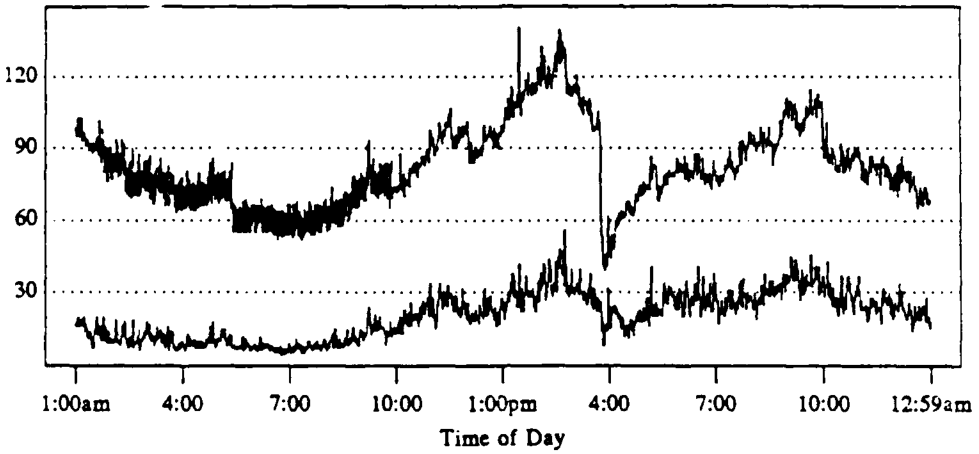
† Since IP and TCP allow options to be inserted in their packet headers, the headers may vary in size. The 20-byte size corresponds to headers without options. The overwhelming majority of TCP packets carry IP and TCP headers without any options.

†† This is not a specification of TCP but an addition of the Berkeley UNIX implementation of the protocol.

5.2.3. Connections

In order to provide data demultiplexing, TCP uses a set of addresses, called *ports*, within each host to address individual processes. The source machine address, the source port number, the destination machine address, and the destination port number uniquely identify a TCP connection. The reliability and flow control mechanisms of TCP require status information for each connection.

FIGURE 5-3. NUMBER OF TCP CONNECTIONS AND ACTIVE TCP CONNECTIONS



The top line in Figure 5-3 shows the total number of TCP connections while the bottom line only shows the number of active TCP connections on our network. Therefore, the segments between the two curves represent idle connections. Each point in the graph corresponds to a minute of network activity. A connection is considered active if at least one data byte is transmitted during the counting interval of one minute; it is considered idle if only acknowledgements or keep-alive packets are transmitted.

Observe how the number of idle connections sharply decreased when, just before 4 pm, ucbarpa, one of the largest VAX's of the Computer Science division, crashed. During the next hour these connections were slowly re-established as workstation users like to have a window connected to a VAX even though they may not use it. (One of the reasons for the high difference in the number of connections before and after the crash, is that ucbarpa is itself a gateway, and is directly connected to the Berkeley Arpanet IMP.) The oscillations that are visible in the hours from about 2 am to 10 am are due to a few ftp connections left open between a LISP machine and several VAX's. We sample every minute while ftp sent keep-alives every two minutes. Notice also how the number of idle connections decreased when a user on a MicroVAX– dikdik– logged out at about 5:20 in the morning. The user had several (idle) connections to machines in the Electrical Engineering department. Finally, notice how the function that represents the number of active TCP connections, unlike that for the total number of connections, resembles the function that describes the network utilization (Figure 4-1).

5.2.4. Higher-Level Protocols

TCP follows the convention of port numbers that are known in the internet environment and to which network servers bind permanently. These well-known port numbers facilitate the procedure for establishing connections. Each server uses a specific higher-level protocol; therefore, by looking at those ports, we can identify the higher protocol traffic components of TCP. Table 5-1, in which the entries are listed in decreasing order of the sum of bytes transmitted from both servers and clients, lists the major components of TCP traffic. All of these protocols can be classified as *application* layer protocols in the OSI model. The DATA columns show the percentage of total data bytes (using the sizes of the TCP data fields) transported by the TCP packets.

TABLE 5-1. TCP PROTOCOL STATISTICS (PERCENTAGES)

PROTOCOL	SERVERS' SIDE			CLIENTS' SIDE		
	PACKETS	BYTES	DATA	PACKETS	BYTES	DATA
login	32.99	21.99	13.61	43.52	20.28	0.42
rcp	3.38	3.73	4.04	4.79	32.89	55.89
printer	0.60	0.32	0.08	0.80	5.57	9.46
ftp	0.40	3.07	5.25	0.42	1.03	1.52
telnet	2.84	1.65	0.72	3.86	1.80	0.05
smtp	0.13	0.08	0.04	0.15	0.14	0.13
Total	40.34	30.84	23.74	53.54	61.71	67.47

The servers' side lists the data that was transmitted by the various network protocol servers (often running on hosts on networks other than XCS), while the clients' side shows the data that user processes sent to servers on XCS or other networks. In some case, as for example for the login server, there is one server per machine. In others, there are only one or a few per local area network such as the case of the printer server.

In Table 5-1 two remote login protocols, login and telnet, and two remote file transfer protocols, rcp and ftp, are listed. This reflects a fundamental trend in, we believe, the majority of local area environments. Even within the scope of a single organization, the need to transmit information and share data on different hardware and software systems, forces system administrators to maintain, and users to employ, a variety of programs

These protocols encompass 93.88 percent of the total number of TCP packets, 92.55 percent of the total TCP bytes, and 91.21 percent of the total TCP data bytes. The remote login protocol is responsible for the majority of packets and bytes sent on the network. This is the combined result of both the multiwindow environment of personal workstations, which enables one to simultaneously open many connections, and the fact that most VAX's are on other networks. Notice, however, that the rcp protocol accounts for most of the data bytes transferred– 59.93 of the total. This again depends on the high number of acknowledgement and keep-alive packets that are sent on idle login connections, whereas rcp is used to copy files.

It is interesting to note that rcp is used primarily to transfer data from the client's side to the server's side (55.98 vs. 4.04 percent of data bytes). We have verified, by identifying sources and destinations of data transferred on the rcp port, that most of the bytes were moved from the SUN workstations to VAX's. Therefore, rcp traffic is primarily generated by remote file copying programs invoked on XCS workstations to "push" files to VAX's, which do not share any file systems with SUN workstations, on networks other than XCS. (The rcp protocol is virtually unused to transfer files between SUN's as the remote file system implementation of SUN UNIX allows workstations to mount file system from various file servers.†)

An analogous situation occurs with the printer protocol: users send files to the various departmental printers located on VAX's. Notice that instead the situation is reversed with the second file transfer protocol: ftp. Its traffic is generated for the most part by users who, from LISP machines, use ftp to both log on to the VAX's and to "pull" files.

Notice that the low-level activity of the mail protocol (smtp) is mostly due to the fact that users tend to send electronic mail or reply to received messages from the machines where they *receive* their mail. Few at Berkeley receive their mail on personal workstations.

† We did observe some rcp traffic between the workstations in the xcssun cluster and xcssun itself.

5.2.5. Source Destination Traffic

Table 5-2 shows the source-destination traffic patterns for the TCP protocol. The table is divided into two parts. Each line on the left side lists the traffic between a source and a destination machine as a percentage of the total TCP traffic. The right side lists the traffic between the same two machines in the opposite direction. The entries are sorted in decreasing order of total bytes transferred between two machines. We use different fonts to represent different classes of machines as we did in Table 4-2: boldface for SUN file servers, roman for diskless SUN workstations, capital for VAX's, and italics for MicroVAX's and LISP machines.

TABLE 5-2. SOURCE-DESTINATION TRAFFIC STATISTICS

SOURCE	DESTINATION	PACKETS	BYTES	SOURCE	DESTINATION	PACKETS	BYTES
sequoia	DALI	0.6130	5.7727	DALI	**sequoia**	0.3066	0.1423
allspice	KIM	0.5809	5.4781	KIM	allspice	0.2942	0.1594
envy-150	KIM	0.4565	4.3613	KIM	envy-150	0.2305	0.1131
ginger	MIRO	0.3836	3.6810	MIRO	ginger	0.2080	0.0959
snow	DALI	0.3694	3.4527	DALI	**snow**	0.1887	0.0870
allspice	DEGAS	0.2876	3.0297	DEGAS	allspice	0.1573	0.0724
xcssun	JI	0.3023	2.7848	JI	**xcssun**	0.1618	0.0746
ginko	INGRES	2.6259	1.4316	INGRES	ginko	1.8324	1.0975
JI	ERNIE	1.7547	1.1105	ERNIE	JI	2.0737	1.0424
shangri-la	INGRES	0.8401	1.7246	INGRES	shangri-la	0.3351	0.1920
uranus	VANGOGH	0.7259	1.2630	VANGOGH	uranus	0.5580	0.5940
paprika	SIM	0.2013	1.7225	SIM	paprika	0.1329	0.0611
CORY	*dikdik*	0.7796	0.9974	*dikdik*	CORY	0.9149	0.5577
JI	CORY	0.8590	0.6941	CORY	JI	1.2368	0.5721
RENOIR	elm	0.6246	0.7984	elm	RENOIR	0.7531	0.3832
VANGOGH	*warthog*	0.1508	1.0918	*warthog*	VANGOGH	0.1051	0.0569
RENOIR	ash	0.9430	0.5779	ash	RENOIR	1.0990	0.5499
CORY	*plato*	0.1765	1.0230	*plato*	CORY	0.1223	0.0977
achilles	KIM	1.1329	0.5206	KIM	*achilles*	0.8124	0.4540
chip	ERNIE	1.2338	0.5673	ERNIE	chip	0.7286	0.3883
RENOIR	redwood	0.6719	0.5741	redwood	RENOIR	0.7750	0.3560
UCBARPA	JI	0.9530	0.4478	JI	UCBARPA	0.6475	0.4097
RENOIR	fir	0.6427	0.4742	fir	RENOIR	0.7628	0.3508
CORY	*aristotle*	0.7748	0.4286	*aristotle*	CORY	0.5669	0.3562

One immediately notices that the imbalance present in Table 4-2, which showed IP traffic, is not present in Table 5-2. No single machine transmits more than six percent of the total TCP byte count. Unlike the transfers of Table 4-2, however, there is a pronounced imbalance, at least in the first few table entries, between entries on the left and right sides of a same lines. (On the first line, for instance, sequoia sends 5.77 percent of the bytes to dali, but receives only 0.14 percent of the bytes from it.) This can be explained by observing that the file system protocols ND and NFS, which account for most of the total IP traffic, generate balanced traffic. In NFS files are read and written, and in ND, which among other things administers the workstation's backing store, paging traffic is first written and then read from remote disks. In the case of TCP instead, file copying, which carries the bulk of the data, goes, for each pair of machines, mainly in one direction.

Observe that all entries, except three, refer to communication between workstations and VAX's– an indication that TCP is mostly used for internetwork traffic.

5.2.6. Interarrival Times

Figure 5-4 shows the interarrival time distribution for the TCP protocol. We compute the interarrival time as the difference between the beginning of the transmissions of two subsequent TCP packets. On the left side of the graphs we list the percentage of the TCP packets, on the right the percentage relative to all packets. The initial peak is due to keep-alive messages and occurs at about 0.4 ms. The peaks numbered 2, 3, 4, and 5 occur after an interval where there are fewer interarrival occurrencies. These peaks correspond to TCP packets that are queued as they are generated during the transmission of one of the many 1K-byte ND packets (see Figure 4-4).

Had we observed the arrivals of packets at the network interface queues, rather than at the network after the Ethernet transmission deference has taken place, we would have seen a distribution closer to that of an exponential distribution.

FIGURE 5-4. PERCENTAGE OF TCP PACKET ARRIVALS (AT RIGHT AS PERCENTAGE OF ALL ARRIVALS)

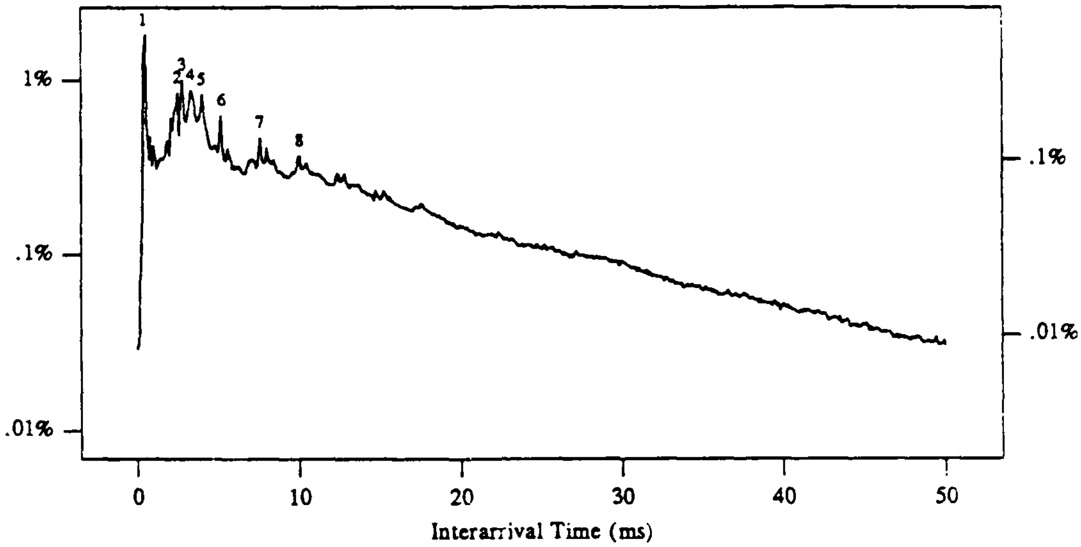
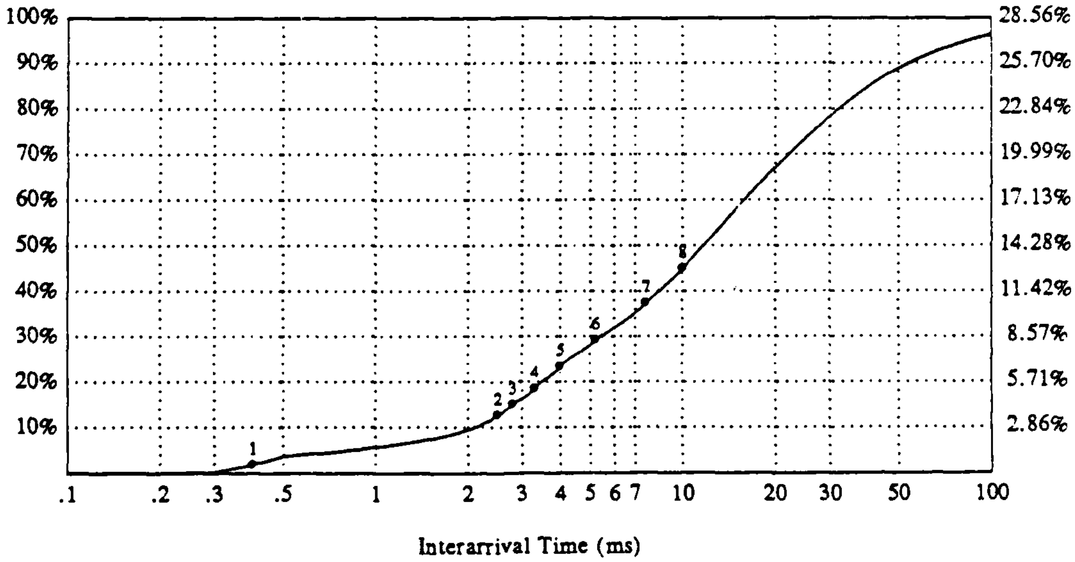


Figure 5-5 plots the cumulative interarrival time distribution for TCP packets. We have marked on the graph the positions of the peaks in Figure 5-4. Notice that 50 percent of the packets are followed by another within 11 ms, 10 percent are followed by another within 2 ms, and 90 percent within 52 ms. TCP is slow compared to ND and NFS, which exclusively carry local traffic; since packet acknowledgements come mainly from machines on other networks, gateways add their switching time to the propagation delays of most TCP packets.

FIGURE 5-5. CUMULATIVE PERCENTAGE OF TCP PACKET ARRIVALS (AT RIGHT AS PERCENTAGE OF ALL ARRIVALS)



5.3. User Datagram Protocol

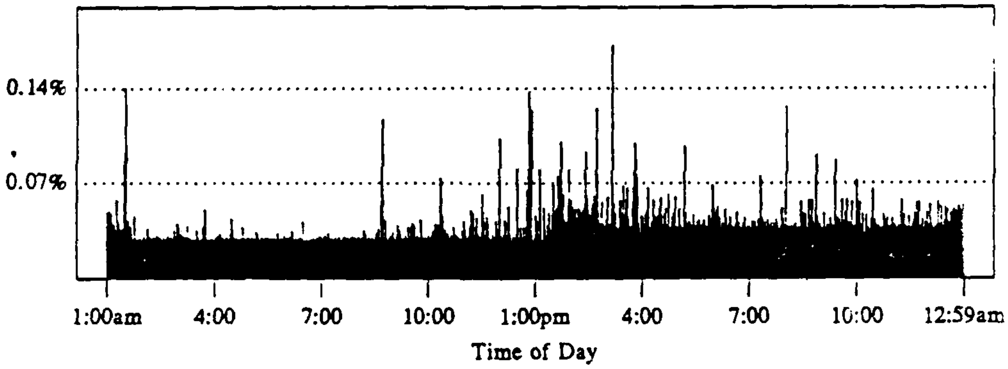
The User Datagram Protocol provides an unreliable, connectionless mechanism for processes to send messages to remote ones. UDP, like TCP, uses IP as its transport protocol and provides data demultiplexing with port numbers. UDP allows broadcasting through the mechanism provided by IP.

UDP is used in the SUN implementation of the Remote Procedure Call protocol, which transports NFS requests. Thus, NFS uses UDP indirectly. Since we have classified and studied NFS independently, in this subsection we look only at the fraction of UDP traffic that is not due to NFS packets.

Although Table 4-1 has already shown that the UDP traffic component is a small fraction of the total XCS traffic, Figure 5-6 shows that it is almost independent of the overall network traffic. A study of the port numbers used by UDP shows that 81.9 percent of the total UDP bytes are sent by the routing daemons that run on each machine and maintain kernel routing tables. These daemons exchange information periodically and the number of packets they send is relatively independent of the traffic.

An analysis of the inter-network traffic for this protocol reveals that 99.7 of the packets remain within the XCS network.

FIGURE 5-6. ETHERNET UTILIZATION (UDP PROTOCOL)



5.4. Summary

IP traffic comprises the great majority of XCS traffic. TCP and UDP account respectively for 28.55 and 16.93 percent of the total packet traffic, or, combined, for about half of the total IP packet traffic. However, most of the UDP traffic is generated by SUN's Network File System Protocol, which we shall analyze as a separate case in Section 7. The remaining part of the UDP traffic, though important for its routing management functions, is statistically negligible.

We observed IP packets transmitted with wrong source or destination IP addresses. They can cause a good deal of trouble in gateways; IP modules should be able to detect these incorrect addresses, and perhaps ignore them, without flooding the network with ICMP error packets.

Most TCP packets are small. They carry character traffic on connections from XCS's workstations to the Computer Science Division's VAX's. Many connections are idle and generate keep-alive packets that result in a large number of 46-byte packets being sent.

The maximum data size carried by a TCP packet is 1K byte. With the new fast machines available today, for which the cost of fragment reassembly is reduced, TCP throughput would increase if larger packet sizes were used.

The TCP interarrival time function approximates the behavior of an exponential distribution.

6. The Network Disk Protocol

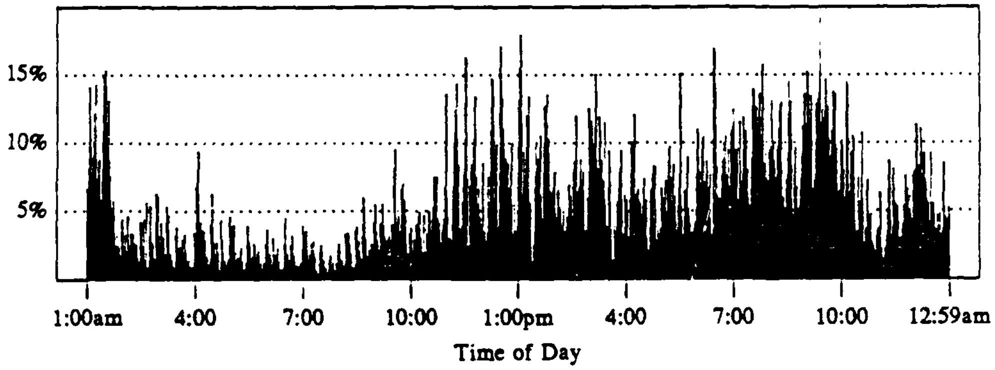
The SUN UNIX operating system allows diskless client workstations to perforn file system activities over the network on remote file servers by using the ND and NFS protocols. The ND protocol is used by a special disk device driver on a diskless workstation when disk I/O operations are requested. This pseudo-device transforms disk block read and write operations into ND messages that are sent to a remote disk server (which normally is also an NFS file server). A server, upon receiving a message, performs the actual disk I/O and sends the response back to the client. Therefore, the ND protocol provides remote disk block access.

The major disadvantage of this type of remote file access is that it does not allow file system write sharing. In fact, since each client stores disk blocks in a locally-maintained buffer cache in order to reduce the number of expensive network accesses, concurrent write operations can create inconsistencies between a client's file data structures and the physical disk image, causing clients to crash.

The NFS protocol provides shared remote file access and thus avoids the problems of the ND protocol. However, some features of the UNIX file system, such as the /dev directory, which contains special files associated with hardware devices, are difficult to implement in a shared, remote file system. Moreover, the performance of NFS, which is implemented with a remote procedure call protocol and is itself a rather complex protocol, is not as good as that of ND.

SUN Microsystems employs both protocols in order to get both performance and flexibility. Whereas NFS provides access to general file systems for which read/write sharing is required, ND is used to implement each client's root file system, a distinct one for each client; to access the paging and swapping area of each workstation; and to access shared portions of the file systems such as /bin that, by containing read-only executable files, do not give rise to inconsistencies. Although there are plans to extend the SUN UNIX system to allow shared root file systems, remote shared swapping, and remote device accesses,[49] *the systems running* in the local area network we measured relied on both NFS and ND.

FIGURE 6-1. ETHERNET UTILIZATION (ND PROTOCOL)



The transport protocol for ND packets is the DARPA IP protocol. Raw IP datagrams have two advantages over UDP datagrams. First, duplication of checksumming is avoided since in IP only the header, and not the entire packet, is checksummed. (Notice that when the physical network medium is the Ethernet, the data-link layer– normally in hardware– checksums the data that it transmits.)† Second, the kernel-level interface to the IP layer is simpler.

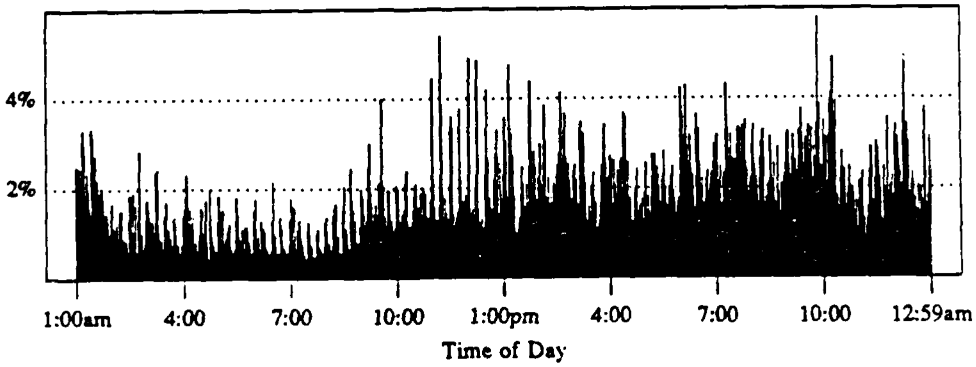
† Our measurements confirm that the Ethernet error rate is on the order of 10^{-6}

6.1. Utilization

Figure 6-1 displays the network utilization attributable to the ND protocol. The graph looks very similar, both in shape and in magnitude, to that of Figure 4-1, which showed the overall Ethernet utilization.

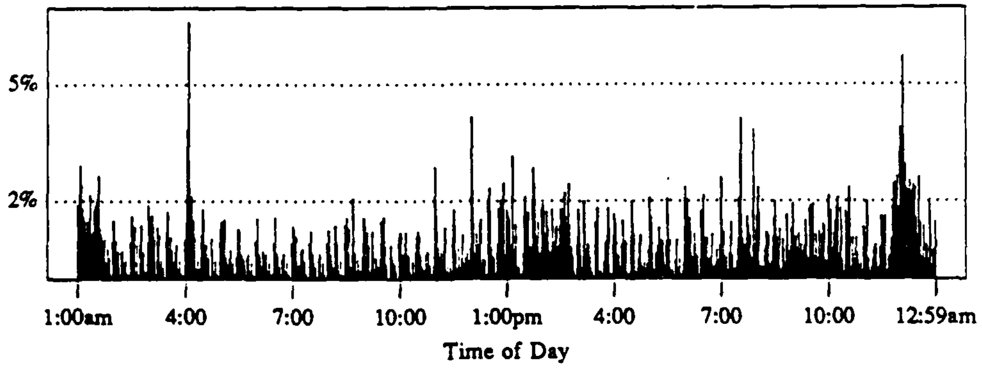
The ND protocol is used to access three different classes of partitions on remote disks: the read/write *private* and *paging* partitions of each individual client, and the read-only *public* partitions, which are shared by client workstations.

FIGURE 6-2. ETHERNET UTILIZATION (ND PROTOCOL: PRIVATE PARTITIONS)



A workstation's ND private partition contains the root file system, the directory /etc (in particular the password file),† the directory /dev (containing the special device files), the /tmp directory, and the mounting points for other directories in the system. Furthermore, files and directories containing information pertaining to each workstation (i.e. varying from system to system), which in UNIX reside on /usr, are, in a diskless SUN environment, symbolic links to the ND private partition. This applies, for instance, to files such as /usr/adm/acct, which collects statistics about the processes that are executed on the system. Therefore, /usr/adm, /usr/spool, and similar directories are symbolic links that point to corresponding directories in the root file system under the directory /private. Accesses to the files in these directories are performed through the ND protocol and generate the network utilization displayed in Figure 6-2.

FIGURE 6-3. ETHERNET UTILIZATION (ND PROTOCOL: PUBLIC PARTITIONS)



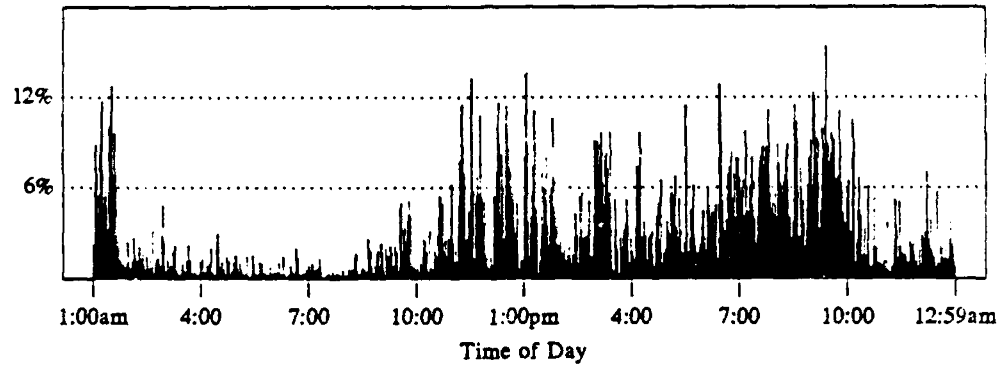
The public partitions provide shared access to programs and libraries so that diskless workstations can save space in their private partitions. This is accomplished by means of a public file

† SUN's yellow page service was not in use on Berkeley's SUN workstations at the time of the measurements.[56]

system at the server that each diskless client "mounts" on the directory /pub. A client then uses symbolic links to access the public files; for instance, the directory /bin would be a symbolic link to the directory /pub/bin. The SUN workstations at UCB were configured so that the public partitions at each file server contained, among other things, the UNIX kernel image, which is loaded into memory at boot time; the directory /bin; and the /lib directory. Figure 6-3 displays the utilization of the Ethernet due to ND protocol from public partitions. The traffic from public partitions is roughly constant throughout the day, suggesting that it is primarily generated by system related tasks– UNIX specific periodic services– independent of users' activity.

The ND protocol is also used by diskless workstations to access their paging partitions to which the virtual memory system of the SUN UNIX kernel directs swapping and paging traffic. In the SUN UNIX virtual memory, processes compete for memory pages from a common page pool. Therefore, if one process requests more memory space and the system free list is empty, the *pagedaemon* looks for pages owned by other processes to free (and writes them to the paging device if dirty). This system generates little swapping activity– the suspension of processes by saving their entire state information on the backing store– but a high amount of paging traffic when workstations' physical memories are not sufficiently large. Figure 6-4 shows the utilization generated by paging activity. Notice how the shape of the graph is similar to that of the overall ND utilization (Figure 6-1), and that of the total Ethernet traffic (Figure 4-1).

FIGURE 6-4. ETHERNET UTILIZATION (ND PROTOCOL: PAGING PARTITIONS)



One might find it surprising to discover that the ND protocol accounts for most of the network traffic. The relative importance of the ND protocol components can be explained by 1) small physical memories on most client workstations, which cause excessive paging activity; 2) small available file buffer caches, which force the kernel to discard pages belonging to commonly used programs; and 3) the tendency of the UNIX operating system to extensively use the directory /tmp, which in the SUN environment resides in the root partition.

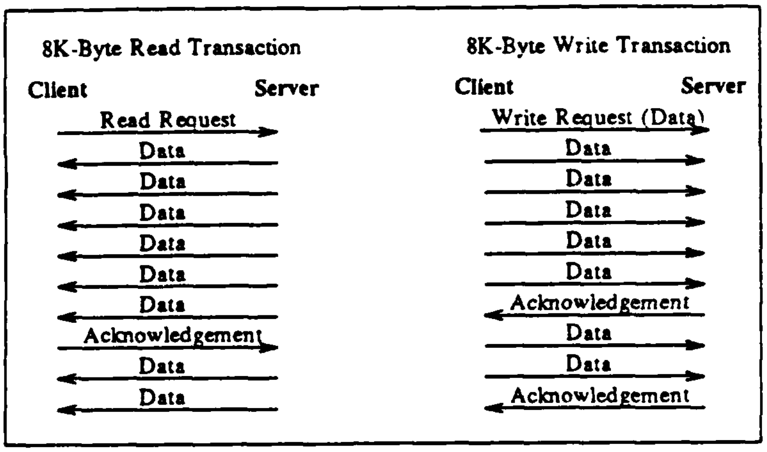
6.2. Packet Length

The packet size distribution for the ND protocol is strictly bimodal. Read-request and acknowledgement packets are small– 48 bytes;† write requests and read responses are 1K-byte packets.

The protocol, which consists only of the two operations read and write, is very simple and is entirely driven from the client side. The data are transferred in 1K-byte packets and workstations or file servers can only send six packets in sequence before being forced to wait for an acknowledgement. Figure 6-5 shows the packet transactions for 8K-byte read and 8K-byte write requests. The client read request generates six reply packets, which the client must acknowledge,

† In most cases SUN3 clients (see Figure 4-4) add two extra empty bytes to the structure that is sent in an ND request, resulting in 50-byte request packets. At times, however, they send 48-byte ND requests.

FIGURE 6-5. NETWORK DISK PROTOCOL TRANSACTIONS



before the server can send the remaining two packets. In the case of an 8K-byte write request, the client first sends six data packets and then must wait for the server to acknowledge before sending the last two packets. (System administrators may change the default number of six, but this was not done at Berkeley at the time of our measurements.)

6.3. Partition Activity

Table 6-1 shows, for each file server, the activity of the ND protocol concerning the private, read-only public, and paging partitions. The third column shows the number of packets directed to a particular partition that were sent or received by a file server. Column four shows the corresponding data bytes transported by those packets.† The following two columns show the number of read operations– each possibly involving more than one packet– that were requested and the number of file bytes that were read. Analogously, the last two columns show the number of write operations performed and the number of bytes written by client workstations to files in the private and paging partitions (there are no writes to the public partition since it is a read-only partition).

One immediately notices the high proportion of paging activity. For sequoia, for instance, it accounts for 67.3 percent of the packets and 68.0 percent of the data bytes. Although the number of read operations to the paging partition is much larger– almost twice as many in the cases of sequoia, xcssun, and lassen– than the number of writes, the difference is not as large for the actual bytes transferred. In fact, in the xcssun, snow, teak, and dim clusters the number of bytes read from the paging partitions is even lower than the number of bytes written. The reason for this behavior is that the maximum read request is limited to 8K bytes, whereas the maximum write request can be as large as 63K bytes. (In the case of the private and public partitions, the maximum request is always 8K bytes.)

Whereas the high level of paging activity certainly depends on the small physical memories in use on our workstations, the fact that virtual memory pages are written to the paging device and rarely, if ever, read back may result from system inefficiencies. Since space allocated for each process on the backing store is organized as a contiguous sequence of blocks, disk transfers are particularly efficient because disk arms need not seek between accesses to a process's data. It is not clear, however, that with today's fast file systems[35] the backing store provides significantly shorter access time. Alternatively, virtual memory could page to and from the file system.[41] This would

† Observe that from this information, by solving a system of two equations in two unknowns, one can derive the number of small and the number of large ND packets. (For SUN3 clients there will be a small error since most [not all] of their request packets are 50 bytes while the servers' acknowledgements are 48 bytes.)

TABLE 6-1. ND PARTITION STATISTICS

FILE SERVER	DISK PARTITION	PACKETS TRANSMITTED	BYTES TRANSMITTED	NO. OF READS	BYTES READ	NO. OF WRITES	BYTES WRITTEN
dim	private	79,616	66,216,960	6,492	17,132,544	12,390	45,262,848
	public	118,691	78,758,544	47,292	73,064,448	0	0
	paging	64,253	54,795,664	7,349	22,284,288	4,995	29,440,000
lassen	private	328,862	281,048,480	6,805	29,558,784	31,521	235,704,320
	public	68,725	57,246,192	6,874	53,947,392	0	0
	paging	61,951	52,087,252	8,055	25,254,912	4,084	23,863,296
sequoia	private	624,707	528,920,186	27,741	141,309,952	53,767	357,571,584
	public	169,585	141,736,620	18,001	133,559,296	0	0
	paging	1,634,613	1,428,094,430	100,009	735,089,664	60,558	614,388,736
snow	private	253,085	208,381,296	26,503	74,147,840	36,869	122,085,376
	public	191,459	165,155,472	22,147	155,968,512	0	0
	paging	378,960	322,268,104	46,897	139,692,032	29,588	164,410,368
teak	private	366,909	311,844,622	12,905	58,514,432	34,415	235,686,912
	public	108,364	93,586,666	11,523	88,346,624	0	0
	paging	414,886	364,721,138	18,936	153,397,248	17,554	191,410,176
xcssun	private	438,281	364,654,000	46,419	134,524,928	59,209	209,091,584
	public	246,215	210,523,472	52,163	198,705,152	0	0
	paging	607,558	515,442,672	76,513	229,871,616	49,914	256,449,536

have the additional advantage, from the point of view of the network traffic, that a process's text pages need not be written to the backing store the first time they are paged out.

It is important to notice that, although there is a monstrous level of paging traffic, due to the way processes compete for each others' pages, SUN UNIX, as well as Berkeley UNIX, generates very little swapping activity. Researchers who have measured Berkeley UNIX or SUN UNIX virtual memory performance and have found little swapping, have often erroneously concluded that virtual memory activity would have a negligible effect on network traffic should network file systems replace local disks.[31, 42]

The careful reader will have noticed that the sum of the packets in the third column of Table 6-1 is 6,156,720 – 1,882 fewer than the number of packets listed in Table 4-1. The missing packets had incorrect IP addresses. Incorrect addresses are not defects of the ND protocol, but a manifestation of the anomalies in the Internet Protocol described in Section 5.1.

6.4. Interarrival Time

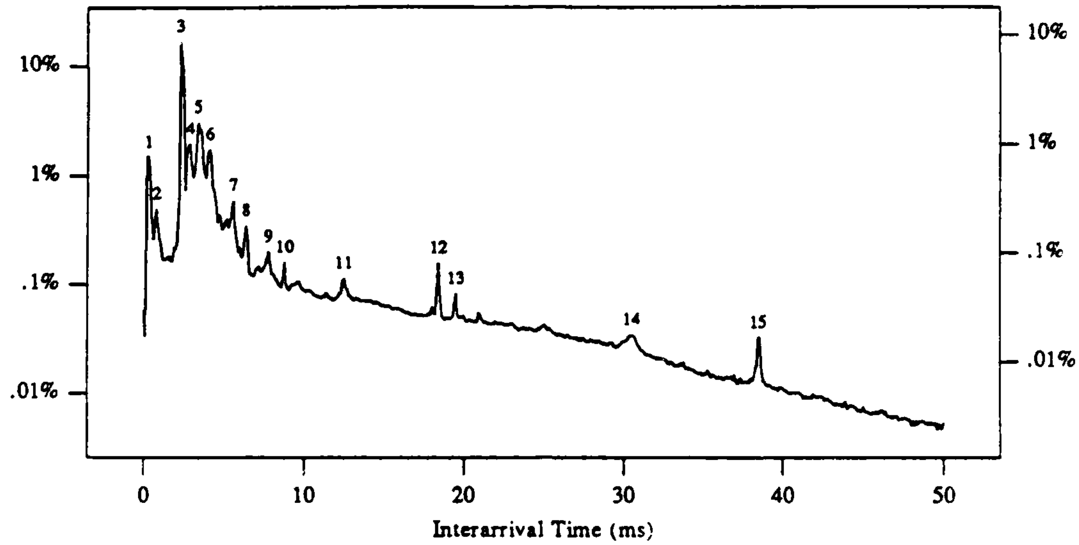
Figure 6-6 is a plot of the packet interarrival time between ND packets. Analogously to what was said in Section 5.2.6, the interarrival time is the difference between the beginning of the transmissions of two subsequent ND packets. Figure 6-7 shows the cumulative interarrival time distribution. In both figures, the vertical axis on the left shows the percentage of ND packets and that on the right the percentage of all packets received.

We have numbered the significant peaks in Figure 6-6 and explained them in Table 6-2. The second column of Table 6-2 lists the client-server pairs whose communication contributed substantially to the peaks listed in the first column. The third column shows the ND partitions to which the traffic was directed; partitions are listed in decreasing order of importance where more than one partition was involved. The fourth column classifies transactions as read or write operations and identifies message sequences, using the names introduced in Figure 6-5, that were responsible for the peaks. The last column shows the percentages of the total peak area that the listed message transactions account for.

These peaks represent patterns in the communication behavior and time constants that are determined by the type of transaction, the disk speed, the CPU speed, and the network interface used. As expected after seeing the influence of the paging activity, the largest component of many of the peaks is the ND paging partition traffic.

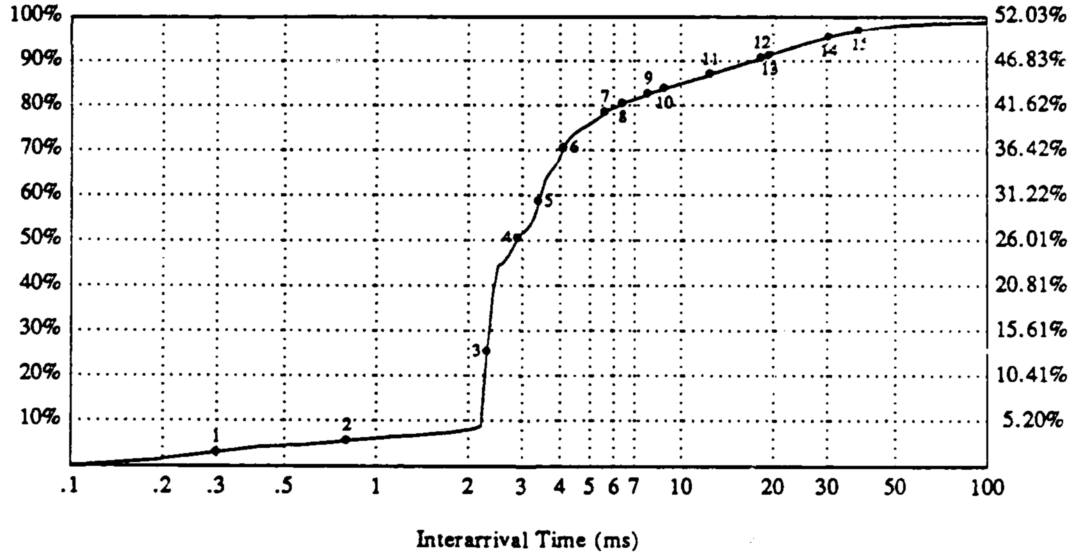
Peaks numbered 1, 3, 5, 6, 8, and 12 are also clearly identifiable in Figure 4-7. This means

FIGURE 6-6. PERCENTAGE OF ND PACKET ARRIVALS (AT RIGHT AS PERCENTAGE OF ALL ARRIVALS)



that no packet belonging to o.her protocols was transmitted between two ND packets in a good portion of arrivals that contributed to these peaks.

FIGURE 6-7. CUMULATIVE PERCENTAGE OF ND PACKET ARRIVALS (AT RIGHT AS PERCENTAGE OF ALL ARRIVALS)



While madrone and sequoia exchanged about 20 percent of the total packet count, as we have seen in Table 4-2, they accounted for more than 40 percent of peaks 1 and 3, for almost 50 percent of peak 2, and for about 35 percent of peak 4. Madrone's and sequoia's dominance was due to the fast processors (Motorola 68020) at both the client workstation and the file server, and their fast network interfaces.

Data/Data message sequences between madrone and sequoia– two SUN3's– account for a large portion of peak 3, which occurs at about 2.2 ms. Since it takes 0.888 ms to transmit a 1K-byte ND packet,† protocol overhead for SUN3's amounts to about 1.3 ms. Some Data/Data

† The ND packet length is 1072 bytes, the Ethernet header is 14-bytes long, the packet's checksum 4-bytes, and there is a 9.6-microsecond interpacket spacing.

message sequences between madrone and sequoia are delayed and have interarrival time of about 3 ms (peak 4). There are three possible reasons for the delay: first, there could be queuing at their network interfaces, which are used by other protocol modules; second, the network channel could be busy when the two machines generate messages for transmission; and third, the two machines could be loaded.

The Data/Data transactions at peak 5, which occurs at around 3.4 ms, are generated mostly by machines in the snow and xcssun cluster. All of these machines are SUN2's; the associated protocol overhead is in this case 2.6 ms.

TABLE 6-2. ND INTERARRIVAL TIME PEAKS

PEAK	MACHINES	PARTITION(S)	MESSAGES	PERCENTAGES
1	madrone-sequoia madrone-sequoia	paging paging	Read Ack/Data Write Ack/Data	22.60 20.56
2	madrone-sequoia madrone-sequoia	paging paging	Read Ack/Data Write Ack/Data	29.13 18.14
3	madrone-sequoia madrone-sequoia ginko-teak	paging paging paging	Read Data/Data Write Data/Data Write Data/Data	23.55 19.97 6.96
4	madrone-sequoia madrone-sequoia madrone-sequoia baobab-lassen	paging paging paging private	Read Data/Data Write Data/Data Read Data/Req Write Data/Data	15.29 13.14 6.11 5.18
5	bashful-snow saturn-xcssun jupiter-xcssun bashful-snow	paging paging/public public/paging paging/private	Read Data/Data Read Data/Data Read Data/Data Write Data/Data	7.78 6.12 6.12 5.65
6	ginko-teak liveoak-sequoia	paging private	Read Data/Data Write Data/Data	16.16 7.63
7	madrone-sequoia madrone-sequoia	paging paging	Write Data/Data Read Data/Data	9.54 6.88
8	madrone-sequoia madrone-sequoia	paging paging	Write Data/Data Write Data/Data	9.89 8.50
9	madrone-sequoia bashful-snow	paging paging	Read Data/Req Write Req/Data	5.95 4.83
10	ginko-teak palm-teak	private/paging private	Write Data/Ack Write Data/Ack	12.60 8.18
11	chip-dim	public	Read Req/Data	53.80
12	chip-dim	public	Read Data/Req	66.55
13	chip-dim	public	Read Data/Req	25.89
14	madrone-sequoia ginko-teak bashful-snow	paging paging paging	Read Req/Data Write Data/Ack Write Data/Ack	9.35 6.68 5.97
15	chip-dim	public	Read Data/Req	71.61

As represented in Figure 6-5, while a Write Data/Data message sequence is generated by client workstations, the Read Data/Data one is generated by servers. The fact that ginko, a client workstation, is a SUN3 and teak, its server, a SUN2, explains why the communication between the two machines shows Write Data/Data sequences that occur mostly at peak 3 and Read Data/Data sequences that occur mostly at peak 6.

The file server dim had a slower non-standard Fujitsu disk, which was exclusively used by the client chip as a public file system partition. The absence of disk contention generates the sharp peaks numbered 11, 12, 13, and 15.

Figure 6-8 displays the interarrival times of ND requests, either read or write requests, issued by client workstations. The graph therefore shows the distribution of times between ND transactions. Table 6-3 gives the main components of the peaks listed in Figure 6-8.

As in Figure 6-6 peak 3 was due to SUN3 machines and peak 5 to SUN2 machines for the

Figure 6-8. Percentage of ND Transaction

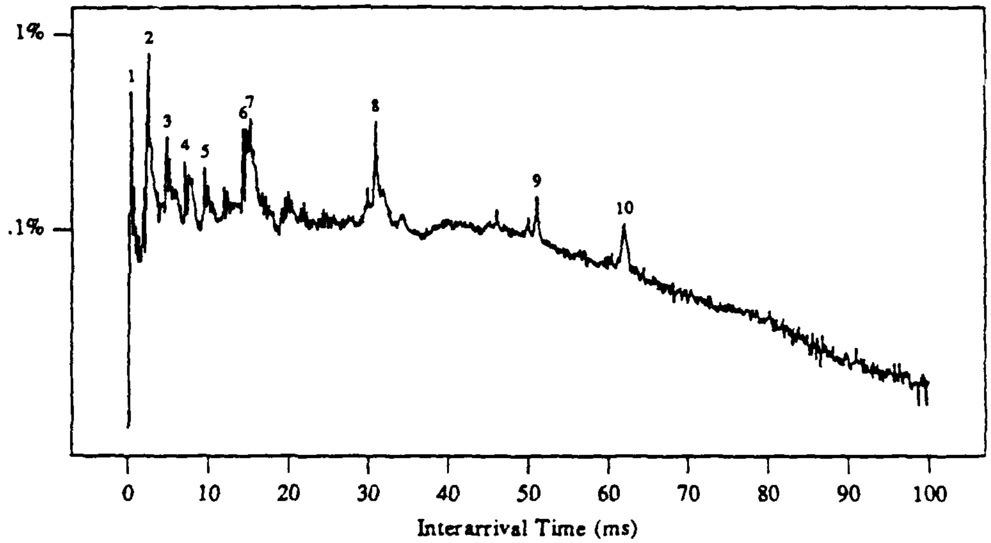


Table 6-3. Interarrival Time Peaks of ND Transactions

Peak	Machines	Partition(s)	Transactions	Percentages
1	madrone-sequoia	paging	Read/Read	22.55
2	jupiter-xcssun	public	Read/Read	4.71
	madrone-sequoia	private	Write/Read	4.71
	saturn-xcssun	public	Read/Read	4.40
	bashful-snow	public	Read/Read	4.35
3	madrone-sequoia	private	Write/Write	15.64
	ginko-teak	private	Write/Write	10.76
	madrone-sequoia	paging	Read/Read	8.23
4	madrone-sequoia	private/paging	Write/Read	31.53
	ginko-teak	private	Write/Write	14.46
5	madrone-sequoia	private/paging	Read/Write	26.70
	bashful-snow	private/paging	Write-Write	9.42
6	madrone-sequoia	private	Write/Write	30.66
	ginko-teak	private	Write-Write	19.46
7	madrone-sequoia	paging	Write-Write	14.60
	jupiter-xcssun	private	Write-Write	11.09
	mars-xcssun	private	Write-Write	9.82
8	chip-dim	public	Read-Read	81.88
9	chip-dim	public	Read-Read	53.48
	madrone-sequoia	paging	Read-Read	19.52
10	madrone-sequoia	paging	Read-Read	15.10
	bashful-snow	paging	Write-Write	10.33
	saturn-xcssun	paging	Write-Write	7.95

same type of Read/Read message sequences, in Figure 6-8 we can see that the time between read transactions is much smaller for SUN3's than for SUN2's (peak 1 at 0.4 ms vs. peak 2 at 2.4 ms). There is no contradiction between these time values and those we derived for the protocol overhead, because client workstations may generate many ND requests in parallel. However, only a few of these requests are transmitted in quick sequence, since file servers cannot sustain high data rates from several client workstations.

6.5. Summary

The Network Disk protocol accounts for most of the network traffic. Paging activity contributes the largest proportion of ND traffic. Three inter-related reasons appear to explain the high level of paging traffic. First, XCS workstations have small physical memories; second, the virtual memory algorithms of SUN UNIX,[2,3] designed when the cost of memory was high compared to the cost of disk space, try to make maximal use of the memory even at the cost of generating some extra I/O traffic; and, third, today's software demands larger virtual address spaces because of its graphic interfaces.

The portion of ND traffic that is directed to the public and private partitions is file system traffic and should be added to that of NFS in quantifying parameters for analytical modeling.

The study of ND interarrival times gave us insights into both protocol overhead and the way machine speed affects response time. The ND protocol is responsible for all of the peaks marked in Figure 4-7, which displayed the interarrival time for all packets in the network.

The protocol is very fast: 90 percent of all ND packets are followed by the next packet within 18 ms. Three factors contribute to this remarkable speed. First, caches– combined with read-ahead policy– work effectively at the server, so that most read operations need not access the disk; second, writes are delayed and a server can send acknowledgements as soon as the data have been received; and third, the protocol is extremely simple, since it uses the IP transport mechanisms to bypass much of the kernel's code for higher-level protocols.

Perhaps the main drawback of ND is the absence of a selective retransmission capability. The loss of a single packet in a multi-packet transfer (we have seen single transactions that transfer as many as 63K bytes!) requires restarting the entire transaction. A second disadvantage is that, while many ND transactions involve the transfer of 8K bytes, the protocol requires that an acknowledgement be sent every 6K bytes. If no acknowledgement were required for an 8K-byte transfer, its transfer time would decrease by roughly ten percent. (One explanation for this six-packet limit is that SUN may have believed that the relatively slow SUN2 machines with the 3Com network interfaces, which have buffer space for only two packets, could not sustain high data rates for long time intervals.) Finally, a third disadvantage of the ND protocol is the way data is organized in a packet. Given that the decision was made to transfer data in multiples of 1K byte, had the protocol information been put at the end of the packet, a benefit of trailer protocols– the data can be page-aligned– would have increased the throughput.

7. The Network File System Protocol

This section analyzes the network performance of the NFS protocol, which provides diskless workstations with remote access to shared file systems over a local area network. It does not directly address the file access properties of SUN remote file system implementation; while the patterns of file access are very important in file server design, they are outside the scope of this study. Understanding the traffic characteristics of the NFS protocol is particularly important because, unlike most of the ND protocol traffic, it represents user rather than system activity.

In order to implement a fully distributed, sharable file system that could overcome the problems that disk block access introduces at the level of ND (ND only allows sharable partitions that are read-only), SUN Microsystems modified the file system data structures. A new interface was designed– the Virtual File System (VFS) interface– that, by providing an additional level of indirection, allows multiple different underlying file system implementations– local or remote– to be accessed uniformly. If a file is local, the VFS layer calls the local device driver, or, in the case of an ND partition, sends a message to the remote disk server. If a file resides on a disk on a remote file server, the VFS layer accesses the Network File System routines with a request that is to be processed remotely. Therefore, the new layer allows file operations at the user level to be transparent as to where the actual file is stored.

7.1. Protocol Layers

The NFS protocol, defined in terms of Remote Procedure Calls, which in the SUN UNIX environment are implemented using the UDP protocol, is responsible for the communication between a client workstation and a file server. Since UDP is an unreliable datagram protocol, the RPC layer handles retransmissions in case of packet loss.

An RPC is an abstraction that extends to a distributed system the well-understood mechanisms for transferring control and data within a computer program using procedure calls.[8,36,57] When a remote procedure is invoked, the calling process is suspended, and the parameters are sent to a remote machine where the procedure is invoked. The results are then sent back and the calling process resumes execution as if returning from a single-machine procedure call. In the SUN UNIX implementation, there are no explicit acknowledgements: the RPC response message is used as an implicit acknowledgement. If no response is received within a short interval from the transmission of the RPC request, the request is sent again.

SUN UNIX has increased the maximum UDP packet size, which was 2K bytes in UNIX 4.2BSD, so that a single RPC can transport 9K bytes of data. For example, in case of a client 8K byte read request, the server fetches the data from the file system and issues a single RPC response. The RPC module on the server side adds an RPC header and passes the request to the UDP module. The UDP module adds a UDP header and passes the data to the IP module. IP takes care of the transmission over the Ethernet channel by fragmenting the message into five 1500-byte packets and one last 920-byte fragment packet. The receiving IP module on the client side reassembles the message, passing it up to the UDP and RPC modules only after the last packet has been received and the message fully reassembled.

The NFS protocol is therefore much more complicated than the ND protocol. Whereas the latter is a specialized protocol designed to move data from a remote disk to the file system cache on a client workstation, the NFS protocol relies on a number of other protocols. Furthermore, NFS, in order to work with heterogeneous file systems, converts the information it transmits into a machine-independent format.[34]

TABLE 7-1. NFS FUNCTIONS

FUNCTION	DESCRIPTION
getattr	Returns the attributes of a file. Used to implement the stat system call.
setattr	Sets the attributes of a file. (mode, uid, gid, size, access time, ...)
lookup	Returns a pointer to a file in a directory. Parses the file's path name one component at a time.
readlink	Returns the string associated with a symbolic link.
read	Reads data from a file.
write	Writes data to a file.
create	Creates a new file.
remove	Removes a file.
rename	Renames a file.
link	Creates a file which is a link to another.
symlink	Creates a symbolic link.
mkdir	Creates a directory.
rmdir	Removes a directory.
readdir	Reads entries in a directory. For efficiency reasons reads many entries in a single call.
statfs	Returns file system information such as block size, number of free blocks, ...

7.2. NFS Procedures

Table 7-1 lis:s the NFS protocol procedures with a brief explanation of their funcuon. There are, among others, procedures to read and write data, to create files, to convert file names into pointers to files, to return file attributes, and to create, remove, or read directory entries. If files are accessed on remote file systems, the kernel's VFS layer transforms the system calls that operate on files into one or more of the remote procedure calls in Table 7-1.

To make the protocol withstand network and file server crashes, NFS file servers are stateless: a server does not maintain state information about any clients. Notice, in fact, that there are no file open and close procedures. As a consequence, path name translation, which converts a file name into a pointer to the file, is performed in NFS by issuing one RPC per path name component. Furthermore, when a procedure returns successfully, the client can assume that the requested operation has been completed and any associated data is on stable storage. If a server waited to write the data (file pages, inode information, and the like) to the disk, the client would have to save those requests in order to resend them to recover from a server crash.

7.3. Utilization

Figure 7-1 shows the network utilization, over one-minute intervals, of the NFS protocol. The pattern of NFS communication generally follows that of the total Ethernet traffic although the graph's spikes indicate that it is more bursty. The one-minute average utilization is well below five percent, with occasional bursts in the range of five to ten percent and one peak at 12 percent after 10 pm.

FIGURE 7-1. ETHERNET UTILIZATION (NFS PROTOCOL)

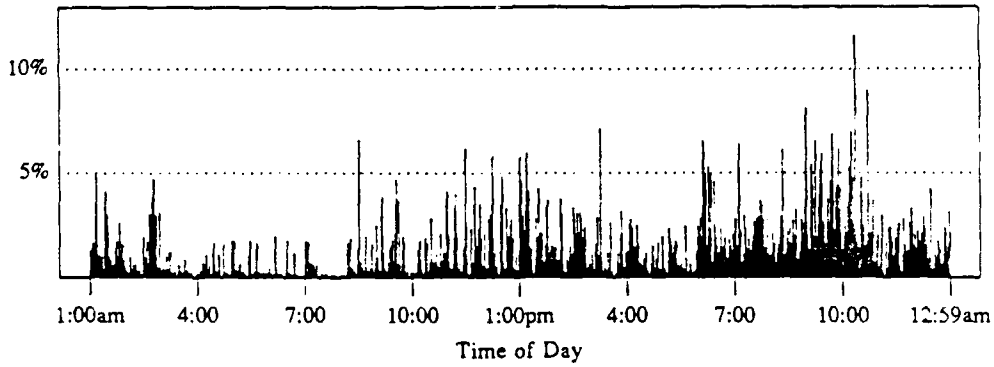


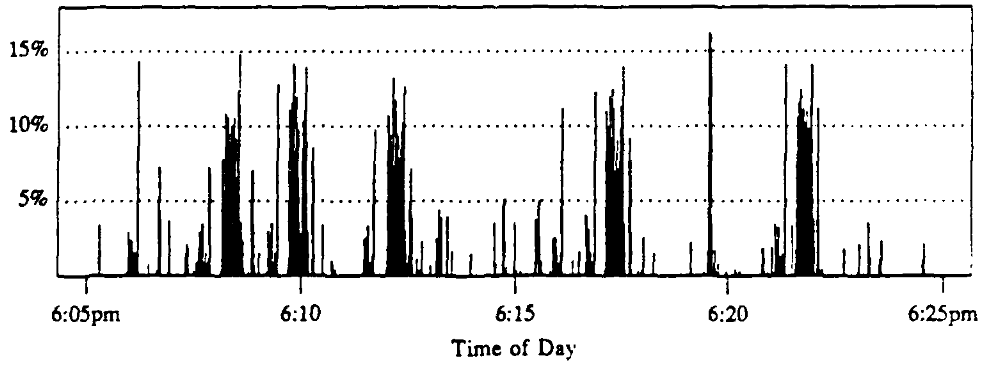
Figure 7-2 displays the network utilization, over one-second intervals, generated by NFS traffic between the client workstation madrone and the file server sequoia. The network activity is intermittent, as it follows the usual pauses in user behavior. In addition to occasional one-second peaks at 15 percent, there are times when the network utilization stays at around 10 percent for several seconds. This corresponds to a throughput rate of about 120K bytes per second. Although used less heavily, other workstations displayed similar data transfer rates over short intervals.

7.4. Packet Length

The packet size distribution generated by the NFS protocol reflects that NFS is based on remote procedure calls. There are a large number of short packets whose lengths are distributed around 144 bytes; these are requests and responses transporting NFS and RPC headers. The data fragmentation performed by the IP module during file read and write operations generates a main peak at 1500 bytes, and minor peaks at 696, 920, and 1276 bytes, which are due to the last fragments of fragmented messages. These peaks are displayed in Figures 4-4 and 4-5.

The distribution of packet lengths is bimodal; however the two modes are different from the

Figure 7-2. Ethernet Utilization (NFS Traffic between Sequoia and Madrone)



modes of ND and TCP.

7.5. NFS Procedure Statistics

Table 7-2 breaks down NFS requests by functions. Three of the NFS routines, mkdir, rmdir, and symlink, are not listed in the table; in one day, we observed only 18, 16, and 3 of these calls, respectively. From this table we derive several observations.

Table 7-2. NFS Functions Statistics

Functic.	Percentage of all NFS Requests
lookup	48.53
read	33.69
readlink	5.35
getattr	4 55
write	4.34
readdir	2.33
create	0.80
remove	0.23
setattr	0.06
rename	0.06
link	0.04
statfs	0.02

Since path name translation is performed one component at a time, lookup operations dominate, representing 48.53 percent of the total NFS requests.

On the average, there are 7.8 read operations for each write operation. A UNIX file system study,[58] in which user requests were traced at the system call interface, reports that there are about 2.7 reads for each write operation. One would expect our read/write ratio to be smaller. not larger, than theirs, since, by looking at the network activity, rather than at the system call activity, we do not capture file read operations that hit the local file caches. We have seen that all write operations go to the server. However, mainly because of their small sizes, our caches are not very effective; thus, the number of reads we observed is close to the number of file system reads issued at the VFS level. Our higher read/write ratio is explained by two factors. First, especially in the paging in of text files, user read requests are large and each request corresponds to more than one NFS packet (recall that page-outs are handled through the ND protocol). Thus, we have a higher number of reads. Second, the accounting information written to the file /usr/adm/acct each time a process terminates is handled in diskless SUN workstations by the ND protocol; therefore, we also have a lower number of writes.

An NFS client caches file data. Repeated use of these data requires a verification from the file server that the data are up to date. This is done with the getattr NFS procedure call. The developers of the NFS protocol discovered that the getattr call accounted for 90 percent of the total number of calls made to the server.[49] The situation was remedied by adding a so-called attribute cache at the client side, which keeps file attributes for 30 seconds after they arrive from the server. The number of getattr procedure calls we observed is 4.55 percent of the total.

There are a large number of readlink operations. Most of them are due to programs accessed in the ND public partitions. For instance, when a user needs to execute the file /bin/ls, the NFS layer issues for the server a lookup request in the root directory with parameter *bin*. The information returned by the server informs the client NFS layer that /bin is a symbolic link. NFS then issues a readlink requests to which the server replies with the name stored in the symbolic link: /pub/bin. At this point, the NFS routines return the information to the VFS layer, since a new path name translation must be started. The VFS layer recognizes that /pub is the mounting point of an ND partition and accesses the file using the ND routines.

The many readlink requests, despite relatively few files in the ND partitions, indicate that client ND caches may not be very effective. This is in accordance with our analysis of the traffic from public ND partitions (Table 6-1). One of the reasons for this ineffectiveness is the large size of frequently used programs and graphics libraries. For instance, libsuntool.a occupies 640K bytes, the C compiler library, libc.a, which is stored in an ND public partition, alone occupies 350K bytes and the compiler commonly requires 1M byte of physical memory.† The size of the UNIX kernel text on a diskless workstation is approximately 500K bytes; SUN UNIX normally allocates ten percent of the total available memory for caches and buffer pools. Therefore, the user of a 4M-byte machine is left with only about 3M bytes of memory for user processes, a good portion of which is taken by SUN's graphics programs. Under these conditions, we should expect both poor cache hit-ratio and heavy paging traffic.

Three of the NFS routines, read, write, and readdir, may transmit in one call more data than fits in a single Ethernet packet. In this case, as we have seen, the IP layer fragments the messages. The average number of packets for the read responses, write requests, and readdir responses is 2.94, 3.11, and 1.56 respectively.

The fact that the number of fragments for the readdir calls is small indicates that most directories that users list contain only few files. Furthermore, since the response time of the UNIX ls command on directories with many entries is particularly slow on diskless workstations, many users avoid listing large directories.

7.6. Protocol Timers

We have observed that an excessive number of NFS requests were retransmitted. Although some of these retransmissions depended on factors not related to the NFS protocol, such as lost packets, many retransmissions were caused by RPC timers expiring before the entire message was received because of delayed packets and delayed server responses. Table 7-3 lists NFS functions and the percentage of their calls that were retransmitted.

Most of these retransmissions occurred 0.7 seconds after the original transmission. Notice that the proportion of write requests that were resent (6.0 percent) is higher than that of read requests resent (1.1 percent). The reason for this is that reads are buffered (i.e. pages are cached at the server, which implements a read-ahead policy) while writes always go to secondary storage.

Figures 7-3 and 7-4 show the number of retransmissions in 12 two-hour intervals for the NFS read and write operations respectively. In the foreground, in white, we show the number of retransmissions; in the middle, in grey, the number of operations; and in the background, in black, the average network utilization in each of the two-hour periods. Although the granularity of the data in these histograms only partially displays the correlation, we have also looked at smaller time intervals. These findings led us to conjecture that both network and disk contention are responsible for these retransmissions.

† These sizes were measured on a SUN3 workstation.

TABLE 7-3. NFS RETRANSMISSIONS

FUNCTION	RETRANSMISSIONS AS PERCENTAGE OF CALLS TO EACH PROCEDURE
write	6.04
read	1.14
readdir	0.57
readlink	0.41
getattr	0.27
lookup	0.19

FIGURE 7-3. PERCENTAGE OF READ RETRANSMISSIONS

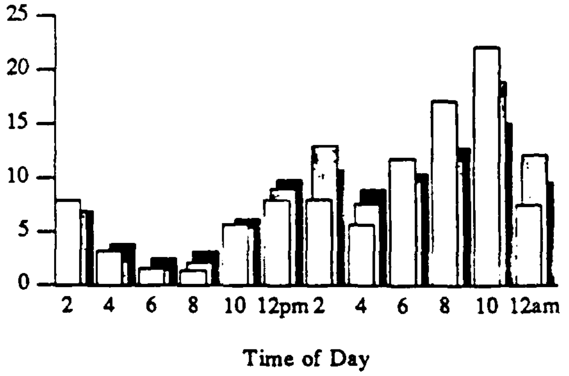
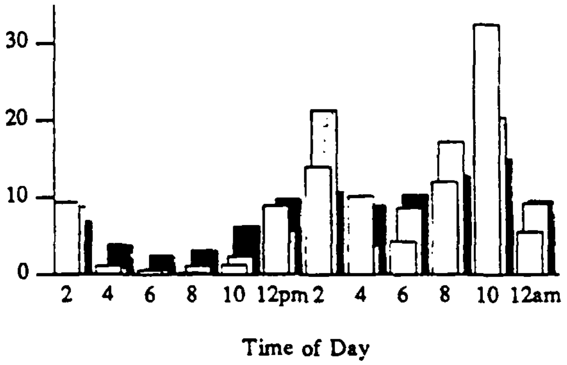


FIGURE 7-4. PERCENTAGE OF WRITE RETRANSMISSIONS



7.7. Interarrival Time

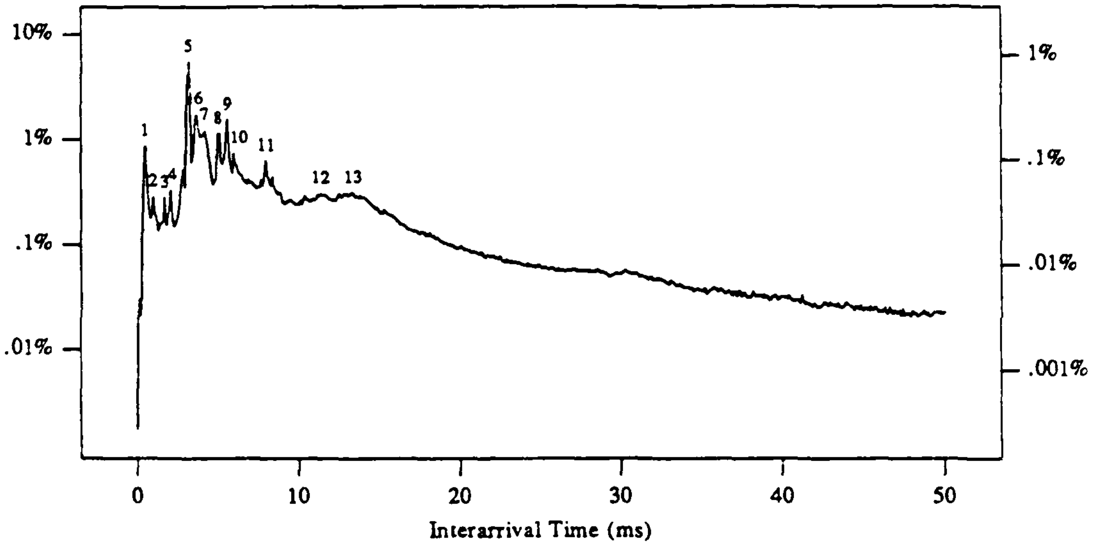
Figure 7-5 displays the NFS packet interarrival time distribution, Figure 7-6 the cumulative interarrival time distribution. In the two figures, the interarrival time is the difference between the beginnings of the transmissions of two subsequent NFS packets. In both figures, the vertical axis on the left shows the percentage of NFS packets and the one on the right the percentage of all packets received. We have numbered the peaks in Figure 7-5, and reported their position in Figure 7-6. Tables 7-4 characterizes the traffic that generated those peaks. For each peak, the table lists the pairs of machines whose communication accounted for a significant proportion of the peak's area. In column three the table shows the type of message sequence that accounted for a portion of the peak. Three different message types are used: *Request*, for an RPC client request; *Response*, for an RPC server response; and *Fragment*, for IP fragments.† The fourth column shows the percentage of the total peak area that each message sequence accounted for.

Again, as in the case of ND, the communication between madrone and sequoia is the largest component of virtually every peak. Furthermore, most of the peaks with lower labels (corresponding to lower interarrival times) are mostly due to SUN3 machines.

The first significant occurrence of Fragment/Fragment message sequences occurs at peak 5 at around 3.1 ms. For the ND protocol and the same pair of machines the Data/Data message exchanges generated interarrival times as low as 2.2 ms. The time difference between these two measures is due in part to the longer messages (50 percent longer in the case of NFS), in part to higher protocol processing time. Using calculations analogous to those of Section 6, we estimate that the protocol overhead amounts to about 1.9 ms for this type of message sequence, for SUN3

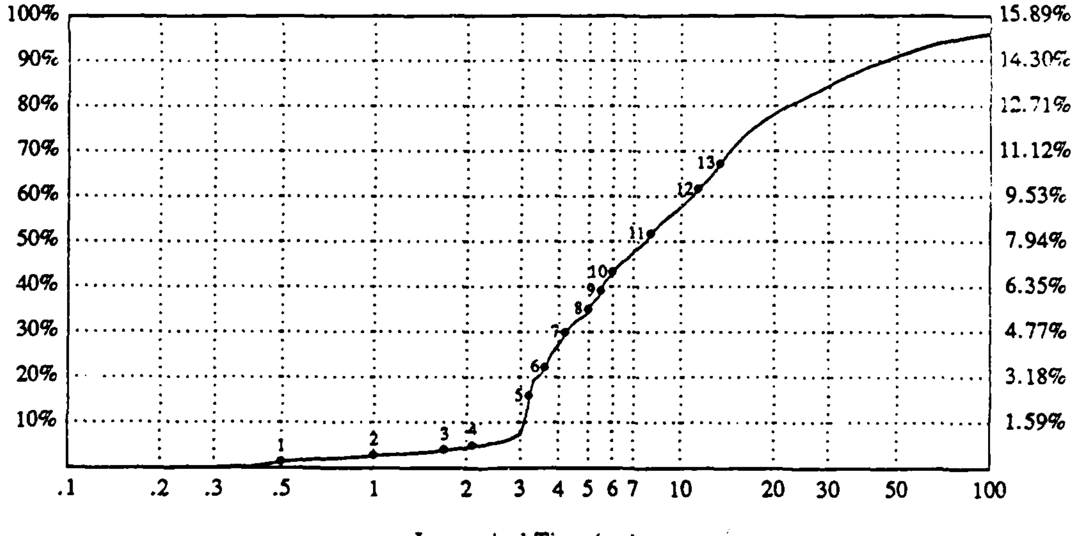
† Recall that the first IP fragment of a fragmented UDP message carries the UDP packet header, while the subsequent fragments only transport the IP packet header. We use the type Fragment only for fragments following the first.

FIGURE 7-5. PERCENTAGE OF NFS PACKET ARRIVALS (AT RIGHT AS PERCENTAGE OF ALL ARRIVALS)



CPU's (it was 1.3 ms for the ND protocol).

FIGURE 7-6. CUMULATIVE PERCENTAGE OF NFS PACKET ARRIVALS (AT RIGHT AS PERCENTAGE OF ALL ARRIVALS)



Peak 7 allows us to compute the equivalent protocol overhead for SUN2 machines. [Notice that the predominant message sequence is a Response/Fragment one, which indicates that SUN2's request fewer data per transaction from file servers. This sequence is otherwise equivalent to a Fragment/Fragment sequence.] The peak occurs at about 4.1 ms; the overhead amounts to about 2.9 ms. (In the case of SUN2 machines the computed ND protocol overhead for Data/Data transactions was 2.6 ms.)

It is important to notice that, although packet protocol overhead is higher in the case of NFS, the increased packet size tends to reduce the per-byte overhead. Since each NFS packet transports 50 percent more data than ND packets (in the case of Data/Data and Fragment/Fragment message sequences), the per-byte protocol overhead is about the same as that of ND: 1.3 microseconds per byte for a SUN3 and 2.2 microseconds per byte for a SUN2. Therefore, the lower NFS protocol throughput must be due to other causes.

TABLE 7-4. NFS INTERARRIVAL TIME PEAKS

PEAK	MACHINES	MESSAGE TRANSACTION	PERCENTAGES
1	madrone-sequoia	Request/Response	9.79
	elm-sequoia	Request/Response	5.41
	shangri-la-sequoia	Request/Response	3.26
2	madrone-sequoia	Request/Response	25.95
	madrone-sequoia	Response/Request	8.63
3	sequoia-shangri-la	Fragment/Request	19.53
	elm-sequoia	Fragment/Request	11.22
	ginko-teak	Fragment/Request	5.87
4	madrone-sequoia	Fragment/Request	17.72
	madrone-sequoia	Fragment/Fragment	10.90
	shangri-la-sequoia	Fragment/Fragment	4.81
5	madrone-sequoia	Fragment/Fragment	46.73
	liveoak-sequoia	Fragment/Fragment	9.85
6	madrone-sequoia	Fragment/Fragment	18.80
	bashful-snow	Fragment/Fragment	8.47
	liveoak-sequoia	Fragment/Fragment	4.46
	ginko-teak	Response/Fragment	4.19
7	bashful-snow	Response/Fragment	14.65
	saturn-xcssun	Response/Fragment	7.58
	doc-snow	Response/Fragment	7.12
8	madrone-sequoia	Request/Response	59.19
	liveoak-sequoia	Request/Response	12.07
	elm-sequoia	Request/Response	4.53
9	madrone-sequoia	Fragment/Fragment	18.71
	madrone-sequoia	Request/Response	15.82
	bashful-snow	Fragment/Fragment	6.06
10	madrone-sequoia	Request/Response	11.39
	bashful-snow	Fragment/Fragment	7.44
	madrone-sequoia	Fragment/Fragment	7.00
	ginko-teak	Fragment/Request	4.37
11	madrone-sequoia	Response/Request	27.36
	bashful-snow	Fragment/Fragment	7.33
	ginko-lassen	Request-Response	4.43

Figure 7-5 shows a slowly decreasing curve, which indicates that NFS operations tend to be more spread over time than the ND ones. There are several reasons for this behavior. First, the NFS ratio of physical disk accesses to number of operations is higher than that of ND; second, while ND's writes are delayed, NFS uses synchronous writes; third, the more complex protocol message interactions in NFS communication create greater time variability; and fourth, NFS remote procedure calls are synchronous, i.e. the client process is blocked until the response is received.

An additional reason for lower performance, one not evident from the traffic measurements, is that NFS client caches are less efficient than ND ones. Their data must be validated before each access. The attribute cache discussed above only alleviates the problem.

7.8. Summary

Interactions with other protocols and its relationship with the Virtual File System layer make the Network File System protocol complex. The protocol is based on remote procedure calls, which, by handling timeouts and retransmissions, provide a reliable transport layer for NFS. RPC timers appear to expire prematurely in the case of long NFS requests, such as 8K-byte read and write operations, leading to unnecessary retransmissions and wasted network bandwidth.

NFS time constants are larger than those of ND. The following numbers capture NFS's longer time constants: 50 percent of the packets are followed by another within 7.2 ms (within 3 ms in the case of ND), while 90 percent of the packets are followed by another packet within 47 ms (within 18 ms for ND). Alternatively, only 35 percent of NFS packets arrive within five milliseconds of each other, compared to 76 percent of ND packets.

However, there is evidence that bulk transfers for the two protocols occur at about the same byte rate. NFS communication between a SUN3 client and a SUN3 server achieves throughput of about 120K bytes per second.

8. Related Work

Although several local area network measurement studies have been performed since the paper by Shoch and Hupp to which we referred in Section 4, only a few have addressed the central issue of this study: how an environment of diskless workstations influences the Ethernet traffic characteristics. We review below some of the works that are most directly related to the point of view we have taken in our analyses.

L. Barnett and M.K. Molloy at the University of Texas at Austin[5] traced the traffic on the campus backbone Ethernet. The network connects some 80 machines, including a number of SUN diskless workstations and their file server. Barnett and Molloy focused on a description of their monitoring facility and on comparisons to other ones, rather than on the analysis and interpretation of the measurements. The accuracy of the study's measurements was limited. It only classified data link layer protocols, in particular the IP protocol. Notably, the authors point out that the local area traffic can be much higher than previously reported (8.7 million packets were observed in one day . However, they fail to identify the reasons for this higher network utilization.

D.R. Cheriton and C.L Williamson[14] measured the traffic properties of an Ethernet at Stanford University tha connects about 50 diskless SUN workstations running the V operating system.[11] A program called *netwatch* was used on a diskless workstation to collect the traffic statistics (in the form of counting events) and another, *timeipc*, to measure protocols' throughput between two workstations. The authors focused on the performance of the VMTP request-response protocol,[13] a general-purpose protocol used for all network communication in V. They did not quantify the network utilization (which was rather low) since they were mainly concerned with the interactive characteristics of their system, such as the message transaction rate, the duration of transactions, and VMTP's retransmission behavior. The measured V systems did not have virtual memory; as a result, most of the traffic was due to file accesses and V's distributed system services, for example, the naming service. Although VMTP is designed to provide higher throughput than current protocols, the authors observed throughput between SUN3's in the range of 100K to 150K bytes, about the same we saw for NFS between madrone and sequoia. They express the opinion that the current generation of protocols is not suitable for future high performance workstations.

E.D. Lazowska et al.,[31] in the context of a modeling study, measured a network of diskless SUN2's and their file servers. Surprisingly, they claim that their workstations, equipped with 2M bytes of physical memory, display a ratio of 4:1 between file access and paging.†

J. Ousterhout et al.[42] studied distributed file system issues using a trace-driven approach. Their traces were collected on large single-machine time-sharing UNIX systems. The authors' main conclusion is that large file caches can dramatically reduce disk I/O traffic; therefore, network bandwidth will not be a limiting factor in building network file systems– "...a network-based file system using a single 10 Mb/s Ethernet can support many hundreds of users [workstations] without overloading the network." They also conclude (citing Lazowska et al.) that paging is rare in UNIX 4.2BSD systems.

In a study on network file system caching in the Sprite operating system, M. Nelson et al.[41] concluded that even without client caching, SUN2 machines require network bandwidths on the

† They measured their system by instrumenting the UNIX kernel rather than by looking at the network and it is conceivable that they might not have captured the entire paging traffic.

order of 80-Kb/s and SUN3's on the order of 250 Kb/s (recall that we observed 1 Mb/s between SUN3's with NFS client-side caching!). In this case the difference can be explained by the fact that they ran benchmarks consisting of compilations, sorting tasks, and text formatting jobs rather than simulating the full workload generated by a user running processes in a window environment. In this work– a follow-up of the trace-driven study of Ousterhout et al.– the authors are more cautious about the Ethernet's ability to sustain the data rate generated by fast machines; nevertheless, they seem more concerned about throughput than about response time.

P.J. Leach et al. have designed an integrated distributed system where diskless workstations and file servers communicate via a 12 Mb/s baseband token passing ring network.[32] Preliminary performance measurements indicate that the average network utilization is low: less than 1/12 of the total network bandwidth was used. However, paging alone consumes ten percent of the network bandwidth when running artificial workload tests. Since in the described implementations the maximum packet size was 550 bytes– their network interfaces allow packets as large as 2048 bytes– higher utilization will be achieved by future, more efficient implementations.

Gonsalves describes measurements on two Ethernets: a 10 Mb/s Ethernet and a 2.94 Mb/s Ethernet.[22] The workload was artificially induced; many computers generated uniformly distributed packets. The packet arrival rate (i.e. the mean of the uniform distribution) was adjusted so as to generate a particular network utilization. Each experiment was repeated with different packet lengths. His main objective was to verify the validity of performance models. Gonsalves found that simple models yield poor results.

The performance of TCP and UDP has been studied by Cabrera et al.[9] They timed the data transferred between user-level processes running on machines on the same network. Therefore, their measurements are user-oriented in that they reflect both network and system overhead. In a second investigation, Cabrera et al.[10] looked closely at the overhead in the UNIX kernel due to TCP and UDP protocol processing. They used a kernel monitor[23] to statistically profile the kernel routines. Both studies seem to indicate that the lowest cost per byte is achieved with a transferred data size of 1024 bytes. This is the reason why UNIX file transfer programs such as rcp and ftp transfer 1K byte of data at a time.

9. Conclusions

In this study we have analyzed the behavior of the Ethernet local area network in an environment of diskless workstations running the UNIX operating system. We have closely compared the traffic characteristics of a medium-size 10-Mb/s Ethernet with those of an older 2.94-Mb/s Ethernet whose traffic was studied at Xerox PARC at the beginning of the decade. This comparison is meaningful since most other local area network traffic studies have not discovered a significant departure from the conclusions of Shoch and Hupp. As a result, system designers currently develop network software and hardware under the implicit assumption that the network load is light.

The most striking conclusion of our study is that diskless workstations may indeed generate enough traffic to substantially load a local area network of the current generation. The communication between a client and a file server generates bursts of activity lasting several seconds that can occupy more than a fifth of the Ethernet's raw bandwidth. The mean network utilization is low (6.5 percent averaged over 24 hours) but bursts generate short-term peak utilization above 30 percent. A system built around diskless workstations is highly interactive, and short-term network utilization, rather than the long-term average, is more likely to affect users' behavior. Therefore, we consider response time more important than throughput in studying the performance of today's systems.

Our measurements show that a workstation's traffic falls into three broad categories: character traffic from the workstation to other machines; paging traffic generated by the workstation's virtual memory to a remote paging device; and file access traffic to remote file servers. A workstation's behavior will depend on the characteristics of each of these three types of traffic. These components were easily identifiable because a different protocol was employed for each component. Character traffic generates many small packets but no substantial network utilization.

Small physical memories and, possibly, sub-optimal performance of the virtual memory algorithms have greatly increased paging traffic. File access to a remote file server generates bursts of traffic lasting several seconds, which may demand bandwidths in the order of 120K bytes per second, or about ten percent of the Ethernet raw bandwidth.

The analyses of the interarrival time for each of the three categories of traffic have revealed that, despite the high packet arrival rate, there are very few occurrences of *back-to-back* packet arrivals– packets whose interarrival time is close to the minimum data link layer interframe spacing of 9.6 microseconds for a 10 Mb/s Ethernet. This is explained by a number of facts. First, there is a high probability that a packet is generated by the same machine as the previous packet. Second, subsequent packets generated on the same machine are mostly due to IP fragmentation in NFS and to protocol-specific fragmentation in ND. Third, as currently implemented by network layer protocols, fragmentation is expensive and even machines as fast as SUN3's cannot keep their network interfaces busy. Based on the previous observations, we argue that, in order to increase the protocols' responsiveness, packet fragmentation should be performed in hardware, or in firmware, on Ethernet interface boards.

Although we have not obtained performance measurements on the number of collisions, we believe that the observed packet arrival times– with spacings among subsequent packet arrivals that correspond almost to a full packet transmission– contribute to keep the number of collisions and protocol latencies small. As network interfaces and protocols are developed that can transmit data, for the same type of user requests, at the rate of the Ethernet data link layer, we would expect that a higher percentage of bandwidth will be wasted in resolving collisions and that protocols' response time will increase. Thus, on the Ethernet, as protocols become more efficient, they must also become more sophisticated to deal with increased latency.

Even though it is difficult to quantify the network bandwidth necessary to support diskless workstations– this is largely a subjective parameter that depends on the performance demanded by users– it seems clear that response time is the single, most important performance characteristic for evaluating these bandwidth requirements. We have seen that bursts of file system traffic may transport data on the order of one to two million bytes. Because of coupling between CPU speed and new application interfaces, future software applications will need to move even greater amounts of data. For this reason, the development of effective caches, as demonstrated by current research efforts and by our measurements, is a very important objective. Yet larger caches will not eliminate users' need for larger amounts of bandwidth. Soon it will be necessary to make precise decisions as to which and how many workstations can be placed on a single-cable Ethernet. Network topology and gateway issues will also become more important.

The Ethernet local area network is a mature product. It provides an extremely effective and low-cost communication medium for interconnecting computers within a local area. Designed when users' communication needs were quite different from today's, the Ethernet has been adapted to provide effective bandwidth to distributed systems based on diskless workstations. Although pushed by improper system configurations on machines with high-performance network interfaces, the Ethernet still functions satisfactorily. Based on the measurements described in this study, we believe that until faster network protocols become available, the Ethernet will provide an effective communication means for workstation users. However, when protocols are developed that are capable of pushing data at the speed of the Ethernet data link layer, a faster technology for local area communication will be necessary.

10. Acknowledgements

Many people contributed to this study. In particular, little would have been accomplished without the support of Domenico Ferrari of the University of California at Berkeley, and Sandy Fraser of the Computing Science Research Center of AT&T Bell Laboratories.

Domenico Ferrari, Mike Karels, and Keith Sklower helped considerably during the measurement phase. Joseph Pasquale and Songnian Zhou also helped during the initial interpretation of the data. Sam Morgan, who read and commented on numerous draft versions of this paper, provided invaluable technical support throughout the analysis phase. Peter Danzig, Domenico Ferrari,

Brian Kernighan, and Dave Presotto commented extensively on the final draft. Among those who also read a draft of the paper, Eric Grosse, Julian Onions, and Howard Trickey provided helpful comments. Finally, Bob Henry, Ed Sitar, and Norman Wilson deserve mention for their help in setting up the various pieces of hardware used in this study. The data analysis required the manipulation of large data sets. The programming language AWK[1] greatly simplified the task. Almost all of the figures in this paper were constructed using GRAP.[7] Brian Kernighan solved all the problems encountered during their preparation.

11. References

1. A.V. Aho, B.W. Kernighan, and P.J. Weinberger, *The AWK Programming Language*, Addison-Wesley (1987).

2. O. Babaoglu, *Virtual Storage Management in the Absence of Reference Bits*, Ph.D. Thesis, Computer Science Division, University of California, Berkeley (Nov. 1981).

3. O. Babaoglu and W.N. Joy, "Converting a Swap-Based System to do Paging in an Architecture Lacking Page-Referenced Bits," *Proc. of the 8th Symposium on Operating System Principles*, pp. 78-86, ACM (1981).

4. R. Ballance, private communication (Aug. 1987).

5. L. Barnett and M.K. Molloy, "ILMON: A UNIX Network Monitoring Facility," *USENIX Conference Proceedings*, Washington, DC, pp. 133-144 (Jan. 1987).

6. F. Baskett, K.M. Chandy, R.R. Muntz, and F.G. Palacios, "Open, Closed, and Mixed Networks of Queues with Different Classes of Customers," *Journal of the ACM* 22(2), pp. 248-260 (April 1975).

7. J.L. Bentley and B.W. Kernighan, "GRAP– A Language for Typesetting Graphs," *Communications of the ACM* 29(8), pp. 782-792 (Aug. 1986).

8. A.D. Birrell and B.J. Nelson, "Implementing Remote Procedure Calls," *ACM Transactions on Computer Systems* 2(1), pp. 39-59 (Feb. 1984).

9. L.F. Cabrera, E. Hunter, M. Karels, and D. Mosher, "A User-Process Oriented Performance Study of Ethernet Networking under Berkeley UNIX 4.2BSD," *Tech. Report No. UCB/CSD 84/217*, University of California, Berkeley, Computer Science Division (Dec. 1984).

10. L.F. Cabrera, M. Karels, and D. Mosher, "The Impact of Buffer Management on Networking Software Performance in Berkeley UNIX 4.2BSD: A Case Study," *Tech. Report No. UCB/CSD 85/247*, University of California, Berkeley, Computer Science Division (June 1985).

11. D.R. Cheriton, "The V Kernel: a Software base for Distributed System," *IEEE Software* 1(2) (April 1984).

12. D.R. Cheriton and T. Mann, *A Decentralized Naming Facility*, Technical Report STAN-CS-86-1098, Computer Science Department, Stanford University (April 1986).

13. D.R. Cheriton, "VMTP: a Transport Protocol for the Next Generation of Communication Systems," *Proc. of SIGCOMM '86*, pp. 406-415, ACM (Aug. 1986).

14. D.R. Cheriton and C.L. Williamson, "Network Measurement of the VMTP Request-Response Protocol in the V Distributed System," Technical Report No. STAN-CS-87-1145, Department of Computer Science, Stanford University (Feb. 1987).

15. Digital Equipment Corporation, Intel Corporation, and Xerox Corporation, *The Ethernet: A Local Area Network Data Link Layer and Physical Layer Specifications*, Version 2.0 (July 15, 1982).

16. Xerox Corporation, *Internet Transport Protocols*, Xerox System Integration Standard, Stamford, CT (Dec. 1981).

17. R.C. Crane and E.A. Taft, "Practical Considerations in Ethernet Local Network Design,"

Proc. of the 13th Hawaii Intl. Conf. on Systems Sci., pp. 166-174 (Jan. 1980).

18. S.E. Deering and D.R. Cheriton, *Host Groups: A Multicast Extension to the Internet Protocol*, Stanford University (Dec. 1985).

19. D.C. Feldmeier, *Empirical Analysis of a Token Ring Network*, Tech. Memo. MIT/LCS/TM-254, M.I.T. Laboratory Computing Science (Jan. 1984).

20. K. Fendick, private communication (Oct. 1987).

21. Gateway Algorithms and Data Structures Task Force, *Toward an Internet Standard Scheme for Subnetting*, RFC 940 (April 1985).

22. T.A. Gonsalves, "Performance Characteristics of two Ethernets: an Experimental Study," *Proc. of SIGCOMM '85*, pp. 78-86, ACM (May 1985).

23. S.L. Graham, P.B. Kessler, and M.K. McKusick, "An Execution Profiler for Modular Programs," *Software – Practice and Experience* 13(8), pp. 671-685 (1983).

24. R. Gusella, *A Study of the Interarrival Time of Network File Access Protocols*, (in preparation).

25. R. Gusella and S. Zatti, "An Election Algorithm for a Distributed Clock Synchronization Program," *IEEE 6th International Conference on Distributed Computing Systems*, Boston (May 1986).

26. R. Jain and S.A. Routhier, "Packet Trains – Measurements and a New Model for Computer Network Traffic," *IEEE Journal on Selected Areas in Communications* SAC-4(6) (Sept. 1986).

27. R. Jain and W.R. Hawe, "Performance Analysis and Modelling of Digital's Networking Architecture," *Digital Technical Journal*(3), Digital Equipment Corporation (Sept. 1986).

28 M.J. Karels, *Another Internet Subnet Addressing Scheme*, University of California, Berkeley (Feb. 1985).

29. M.J. Karels, private communication (Dec. 1986).

30. S.S. Lam, "A Carrier Sense Multiple Access Protocol For Local Networks," *Computer Networks* 4(1), pp. 21-32 (Feb. 1980).

31. E.D. Lazowska, J. Zahorjan, D.R. Cheriton, and W. Zwaenepoel, "File Access Performance of Diskless Workstations," *ACM Transactions on Computer Systems* 4(3), pp. 238-270 (Aug. 1986).

32. P.J. Leach, P.H. Levine, B.P. Douros, J.A. Hamilton, D.L. Nelson, and B.L. Stumpf, "The Architecture of an Integrated Local Network," *IEEE Journal on Selected Areas in Communications* 1(5), pp. 842-857 (Nov. 1983).

33. S.J. Leffler and M.J. Karels, *Trailer Encapsulation*, RFC 893, University of California, Berkeley (April 1984).

34. B. Lyon, *SUN External Data Representation Specification*, SUN Microsystems (1984).

35. M.K McKusick, W.N. Joy, S.J. Leffler, and S.J. Fabry, "A Fast File System for UNIX," *ACM Transactions on Computer Systems* 2(3), pp. 181-197 (Aug. 1984).

36. SUN Microsystems, "Remote Procedure Calls User's Manual," *UNIX Documentation*.

37. SUN Microsystems, *ND(8)*, UNIX User's Manual, Section 8 (1983).

38. SUN Microsystems, *Network File System Protocol*, Protocol Specifications (1984).

39. J. Mogul, *Internet Subnets*, RFC 917, Stanford University (Oct. 1984).

40. J. Mogul and J. Postel, *Another Internet Subnet Addressing Scheme*, RFC 950, USC Information Sciences Institute (Aug. 1985).

41. M. Nelson, B. Welch, and J. Ousterhout, "Caching in the Sprite Network File System," *Report No. UCB/CSD 87/345*, University of California, Berkeley, Computer Science Division (March 1987).

42. J. Ousterhout, H. Da Costa, D. Harrison, J. Kunze, M. Kupfer, and J. Thompson, "A Trace-Driven Analysis of the UNIX 4.2BSD File System," *Proc. of 10th ACM Symposium on*

Operating System Principles, Orcas Island, WA (Dec. 1985).

43. D.C. Plummer, *An Address Resolution Protocol*, RFC 826, Massachussets Institute of Technology (Nov. 1982).

44. J. Postel (ed.), *Internet Control Message Protocol*, RFC 792, USC Information Sciences Institute (Sept. 1981).

45. J. Postel (ed.), *Transmission Control Protocol*, RFC 793, USC Information Sciences Institute (Sept. 1981).

46. J. Postel (ed.), *Internet Protocol – DARPA Internet Program Protocol Specification*, RFC 791, USC Information Sciences Institute (Sept. 1981).

47. D. Presotto, private communication (July 1987).

48. A.P. Rifkin, M.P. Forbes, R.L. Hamilton, M. Sabrio, S.Shah, and K. Yueh, "RFS Architectural Overview," *USENIX Conference Proceedings*, Atlanta, pp. 248-259 (June 1986).

49. R. Sandberg, D. Goldberg, S. Kleinman, D. Walsh, and B. Lyon, "Design and Implementation of the Sun Network Filesystem," *Usenix Conference Proceedings*, pp. 119-130 (June 1985).

50. R.W. Scheifler and J. Gettys, "The X Window System," *ACM Transactions on Graphics*(63) (1986).

51. J.F. Snoch and J.A. Hupp, "Measured Performance of an Ethernet Local Network," *Communications of the ACM* 23(12), pp. 711-721 (December 1980).

52. J.F. Shoch, Y.K. Dalal, D.D. Redell, and R.C. Crane, "The Ethernet," pp. 549-572 in *Local Area Networks: An Advanced Course*, ed. D. Shepherd, Springer-Verlag (1982).

53. J. Sventek, W. Greiman, M. O'Dell, and A. Jansen, "A Comparison of Experimental and Theoretical Performance," *Computer Networks*(8), pp. 301-309 (August 1984).

54. D. Swinehart, G. McDaniel, and D. Boggs, "WFS: A Simple Shared File System for a Distributed Environment," *Proc. 7th ACM Symposium on Operating System Principles* (Dec. 1979).

55. K. Thompson, "UNIX Time-Sharing System: UNIX Implementation," *Bell System Technical Journal* 57(6), pp. 1931-1946 (July-Aug. 1978).

56. P. Weiss, *Yellow Pages Protocol Specification*, Sun Microsystems (1985).

57. B.B. Welch, "The Sprite Remote Procedure Call System," *Report No. UCB/CSD 86/302*, Berkeley, Computer Science Division, University of California (June 1986).

58. S. Zhou, H. Da Costa, and A.J. Smith, "A File System Tracing Package for Berkeley UNIX," *Report No. UCB/CSD 85/235*, University of California, Berkeley, Computer Science Division (May 1985).